FIG. 1A

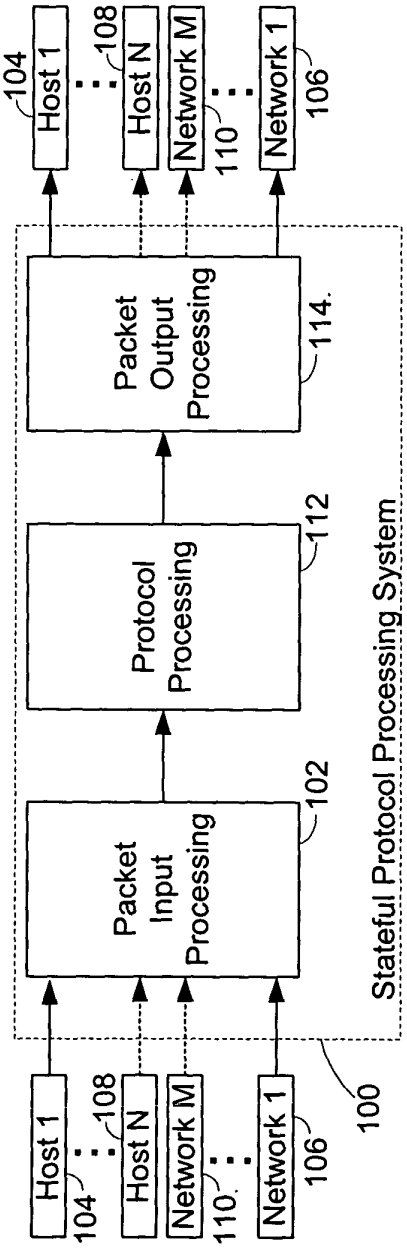
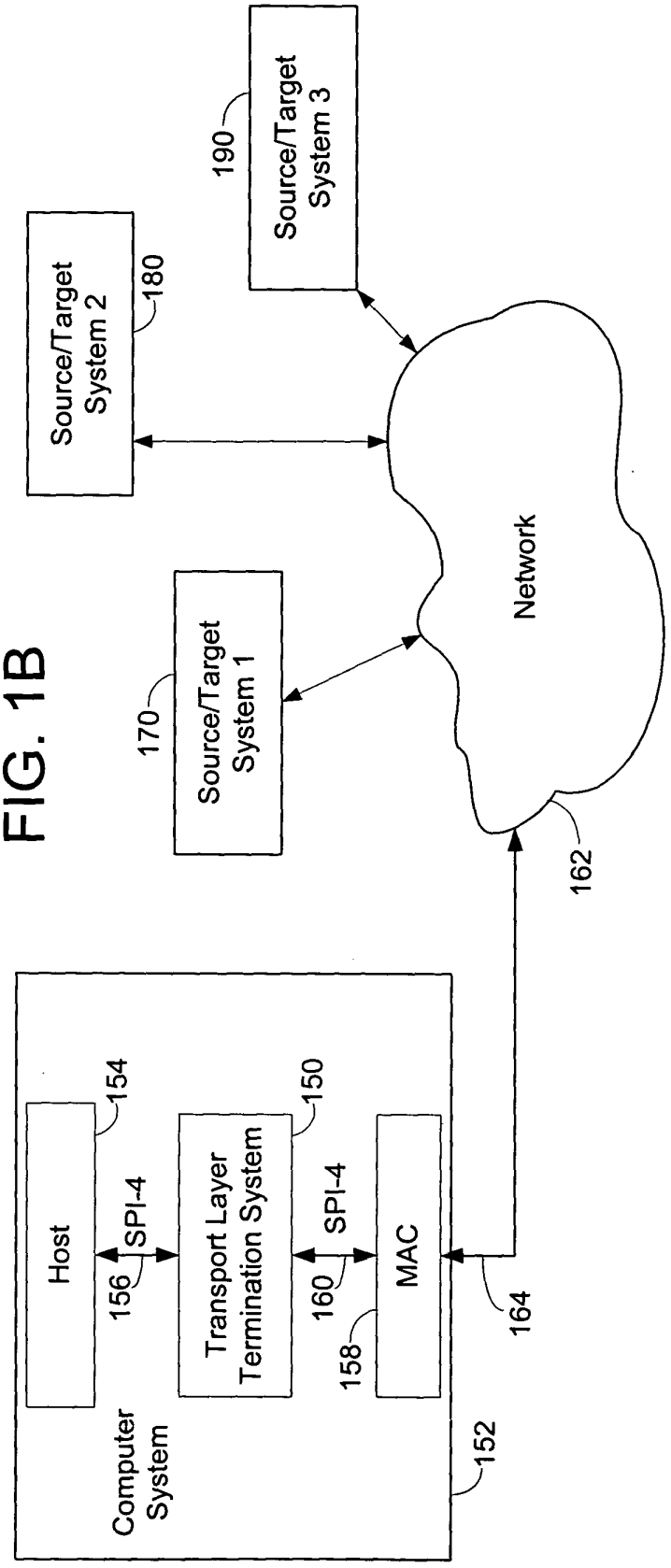

Host 1
104
Host N
108
Network M
110
Network 1
106
Packet Input Processing
102
Protocol Processing
112
Packet Output Processing
114
Stateful Protocol Processing System
100

FIG. 1B


Source/Target System 1
170
Source/Target System 2
180
Source/Target System 3
190
Network
162
Computer System
Host
154
SPI-4
156
Transport Layer Termination System
150
SPI-4
160
MAC
158
152
164

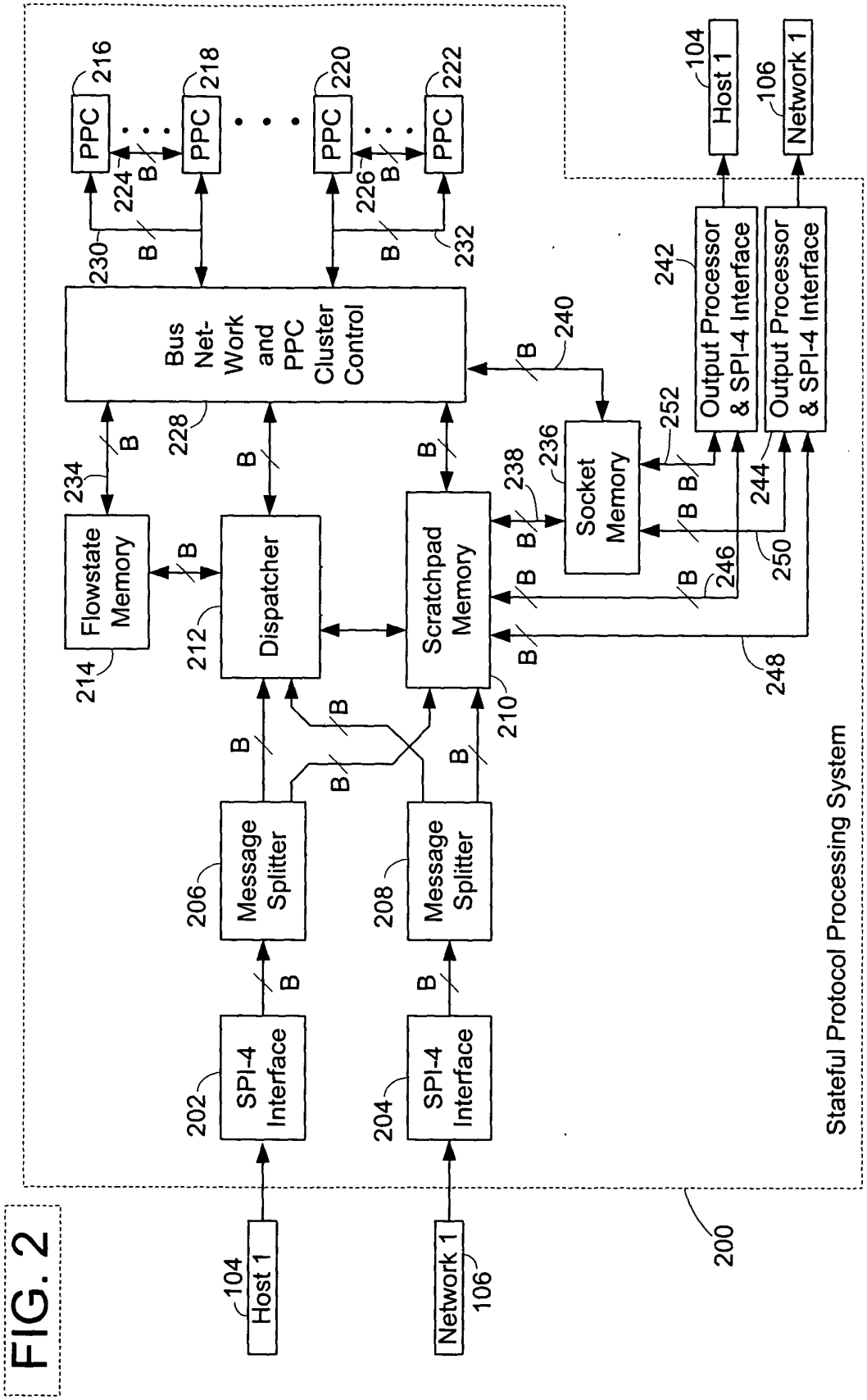

Stateful Protocol Processing System
200
Host 1
104
Network 1
106
SPI-4 Interface
202
SPI-4 Interface
204
Message Splitter
206
Message Splitter
208
Dispatcher
212
Flowstate Memory
214
Scratchpad Memory
210
Socket Memory
236
Bus Net-Work and PPC Cluster Control
228
PPC
216
PPC
218
PPC
220
PPC
222
Output Processor & SPI-4 Interface
242
Output Processor & SPI-4 Interface
244
B
224
226
230
232
234
238
240
246
248
250
252

FIG. 2

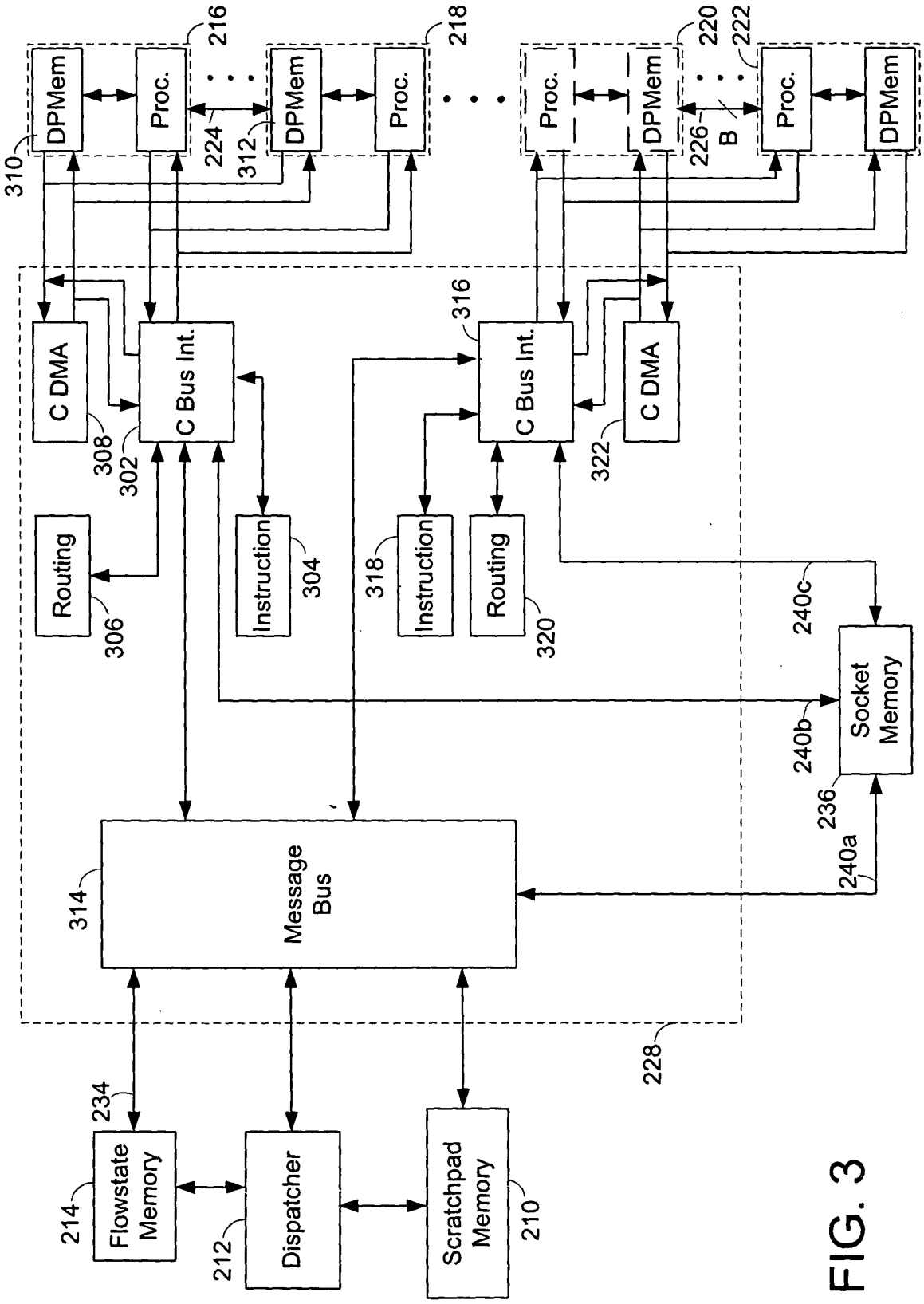

DPMem
Proc.
216
218
220
222
224
312
226
B
310
C DMA
C Bus Int.
308
302
316
322
Routing
306
Instruction
304
318
320
240c
Socket Memory
240b
236
240a
Message Bus
314
228
234
Flowstate Memory
214
Dispatcher
212
Scratchpad Memory
210

FIG. 3

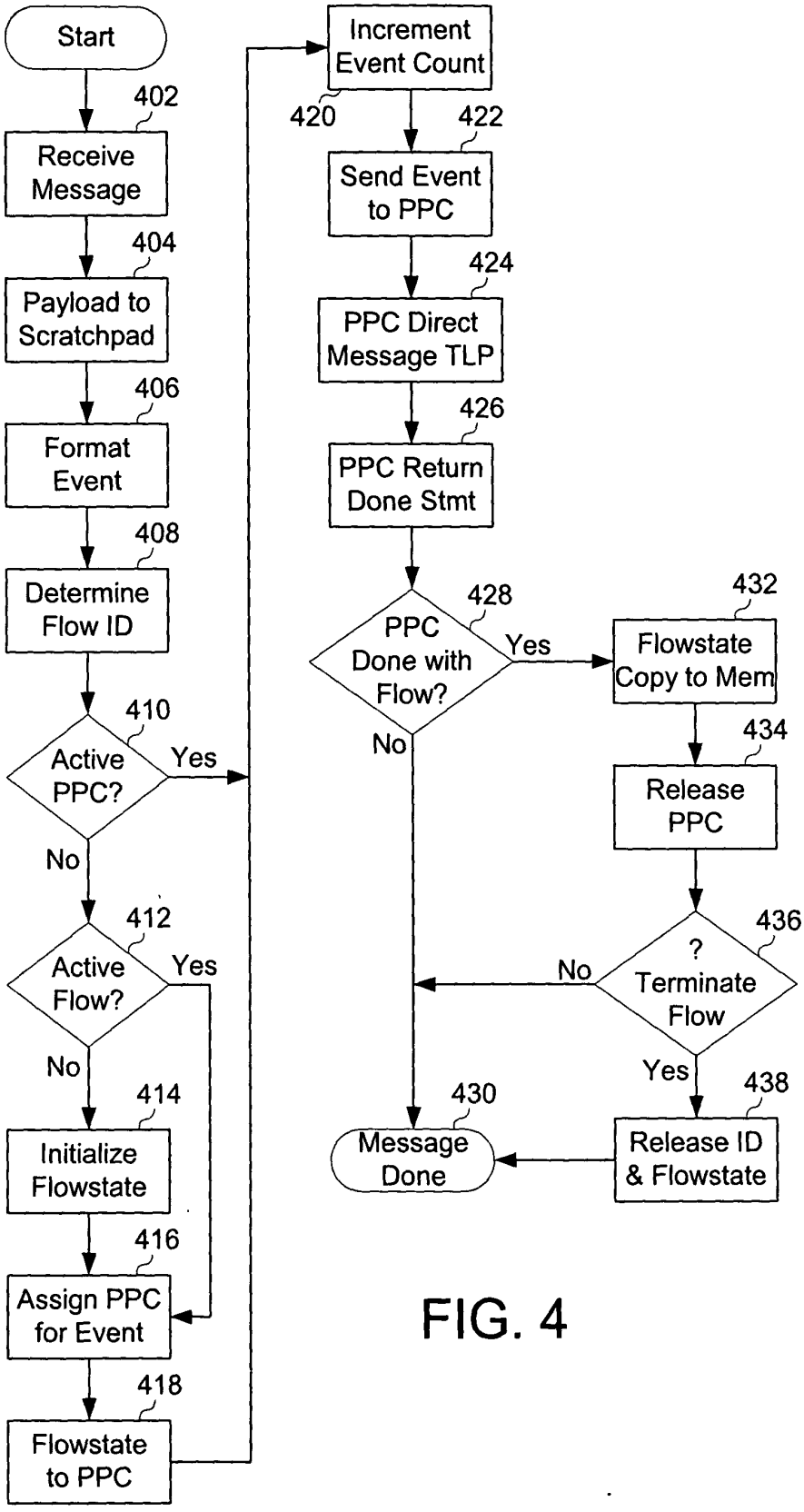

Start
402
Receive Message
404
Payload to Scratchpad
406
Format Event
408
Determine Flow ID
410
Active PPC?
Yes
No
412
Active Flow?
Yes
No
414
Initialize Flowstate
416
Assign PPC for Event
418
Flowstate to PPC
Increment Event Count
420
422
Send Event to PPC
424
PPC Direct Message TLP
426
PPC Return Done Stmt
428
PPC Done with Flow?
Yes
No
432
Flowstate Copy to Mem
434
Release PPC
436
? Terminate Flow
No
Yes
438
Release ID & Flowstate
430
Message Done

FIG. 4

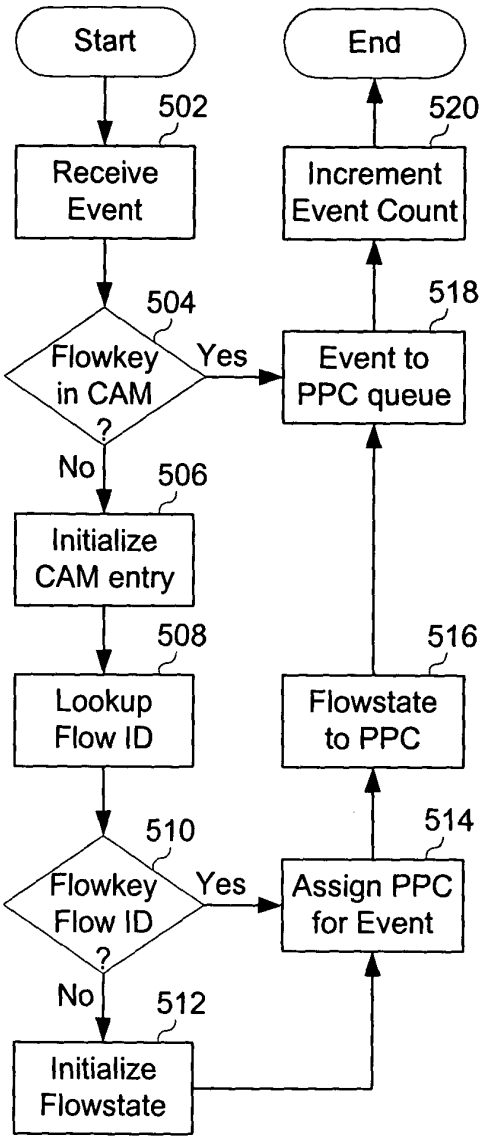
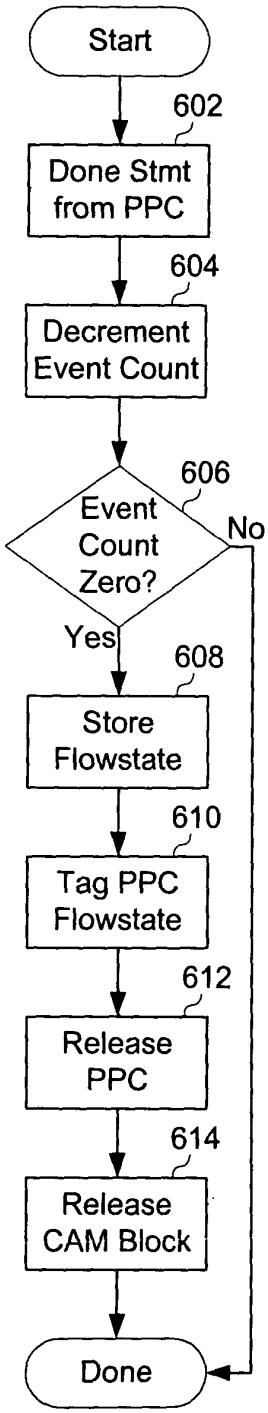

Start
End
502
Receive Event
520
Increment Event Count
504
Flowkey in CAM ?
Yes
518
Event to PPC queue
No
506
Initialize CAM entry
508
Lookup Flow ID
516
Flowstate to PPC
510
Flowkey Flow ID ?
Yes
514
Assign PPC for Event
No
512
Initialize Flowstate

FIG. 5

FIG. 6


Start
602
Done Stmt from PPC
604
Decrement Event Count
606
Event Count Zero?
No
Yes
608
Store Flowstate
610
Tag PPC Flowstate
612
Release PPC
614
Release CAM Block
Done

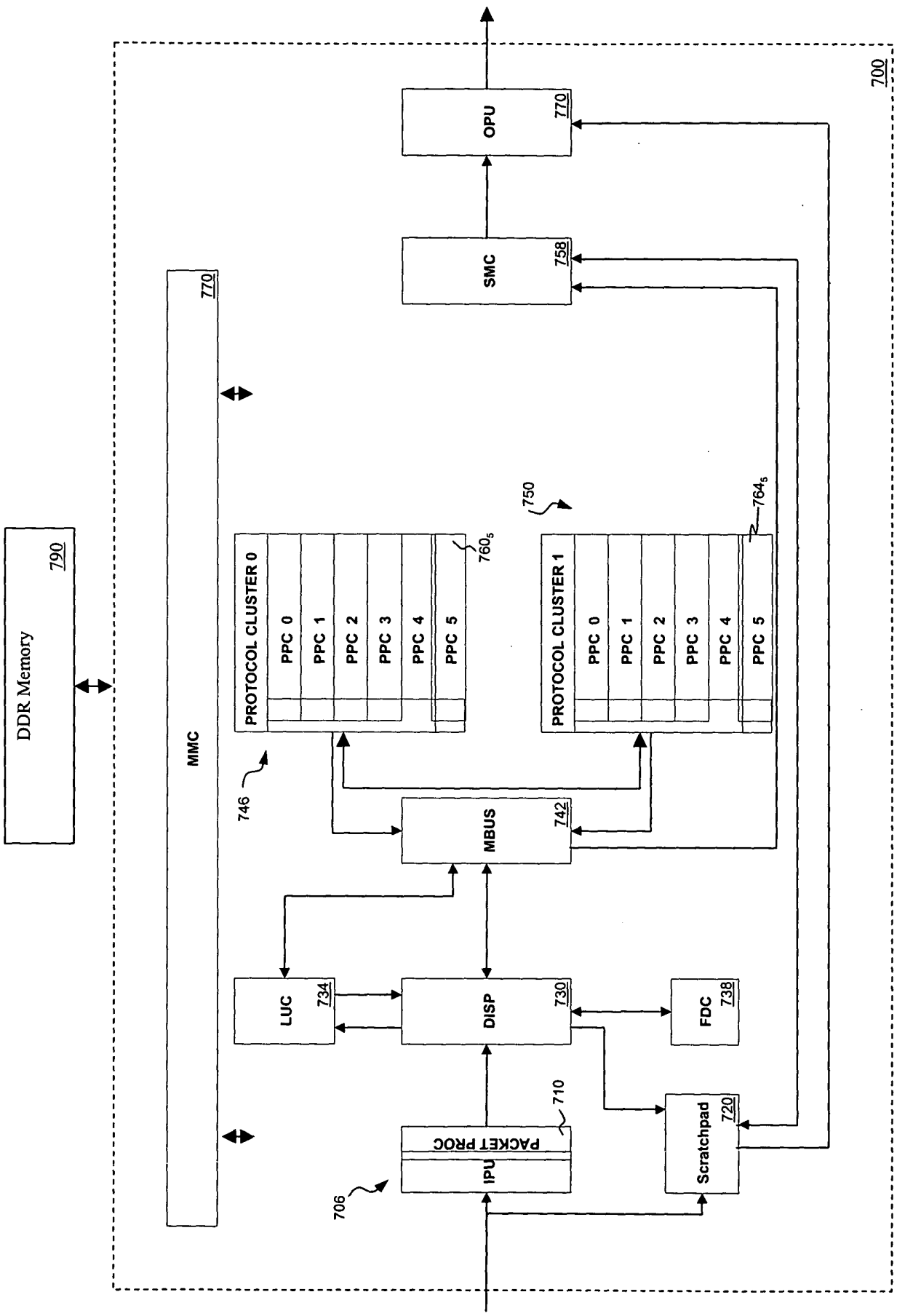

700
OPU
770
SMC
758
770
MMC
DDR Memory
790
750
764s
760s
PROTOCOL CLUSTER 0
PPC 0
PPC 1
PPC 2
PPC 3
PPC 4
PPC 5
PROTOCOL CLUSTER 1
PPC 0
PPC 1
PPC 2
PPC 3
PPC 4
PPC 5
746
MBUS
742
LUC
734
DISP
730
FDC
738
710
PACKET PROC
IPU
706
Scratchpad
720

FIGURE 7

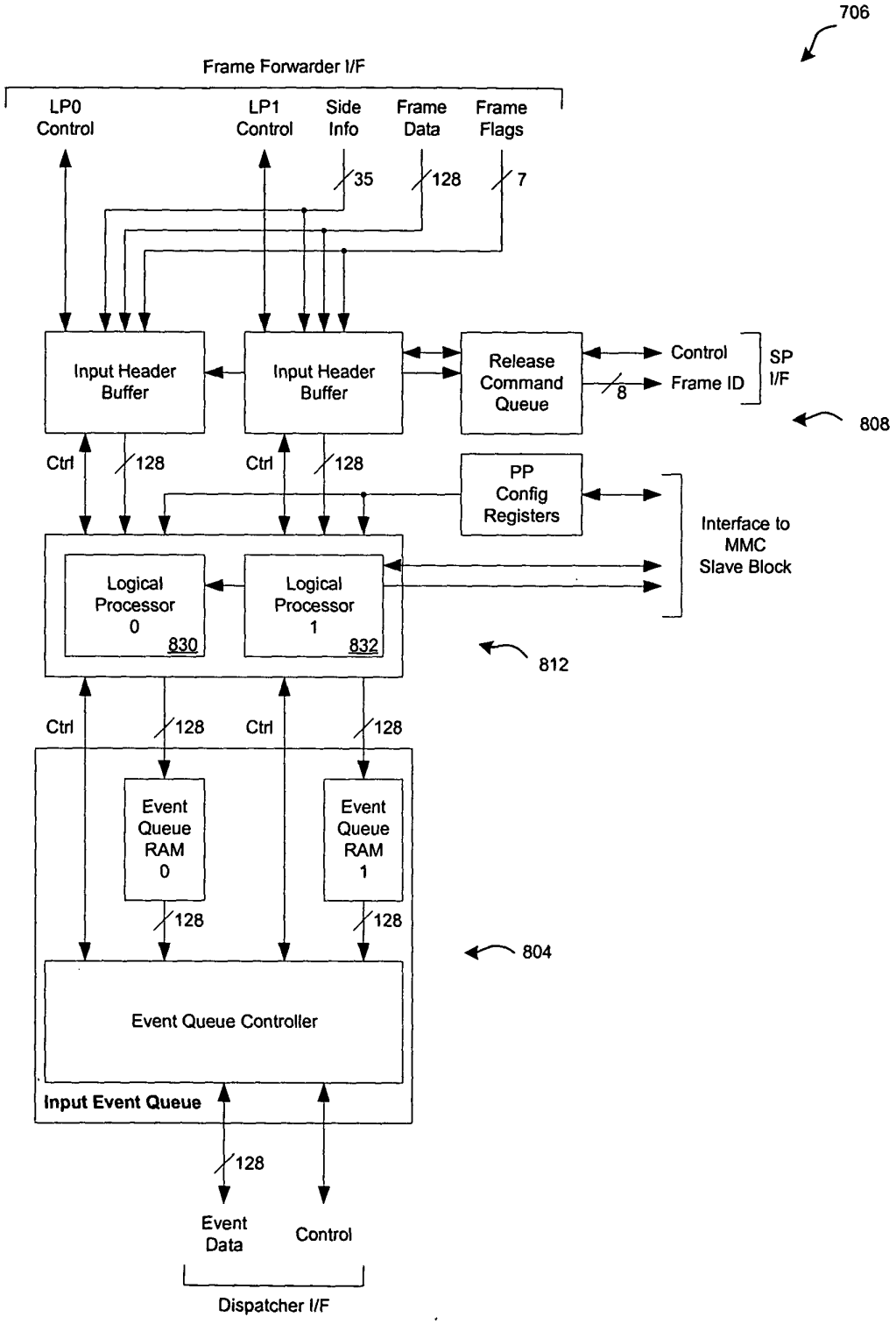

706
Frame Forwarder I/F
LP0 Control
LP1 Control
Side Info
Frame Data
Frame Flags
35
128
7
Input Header Buffer
Input Header Buffer
Release Command Queue
Control
Frame ID
8
SP I/F
808
Ctrl
128
Ctrl
128
PP Config Registers
Interface to MMC Slave Block
Logical Processor 0
830
Logical Processor 1
832
812
Ctrl
128
Ctrl
128
Event Queue RAM 0
Event Queue RAM 1
128
128
804
Event Queue Controller
Input Event Queue
128
Event Data
Control
Dispatcher I/F

FIGURE 8

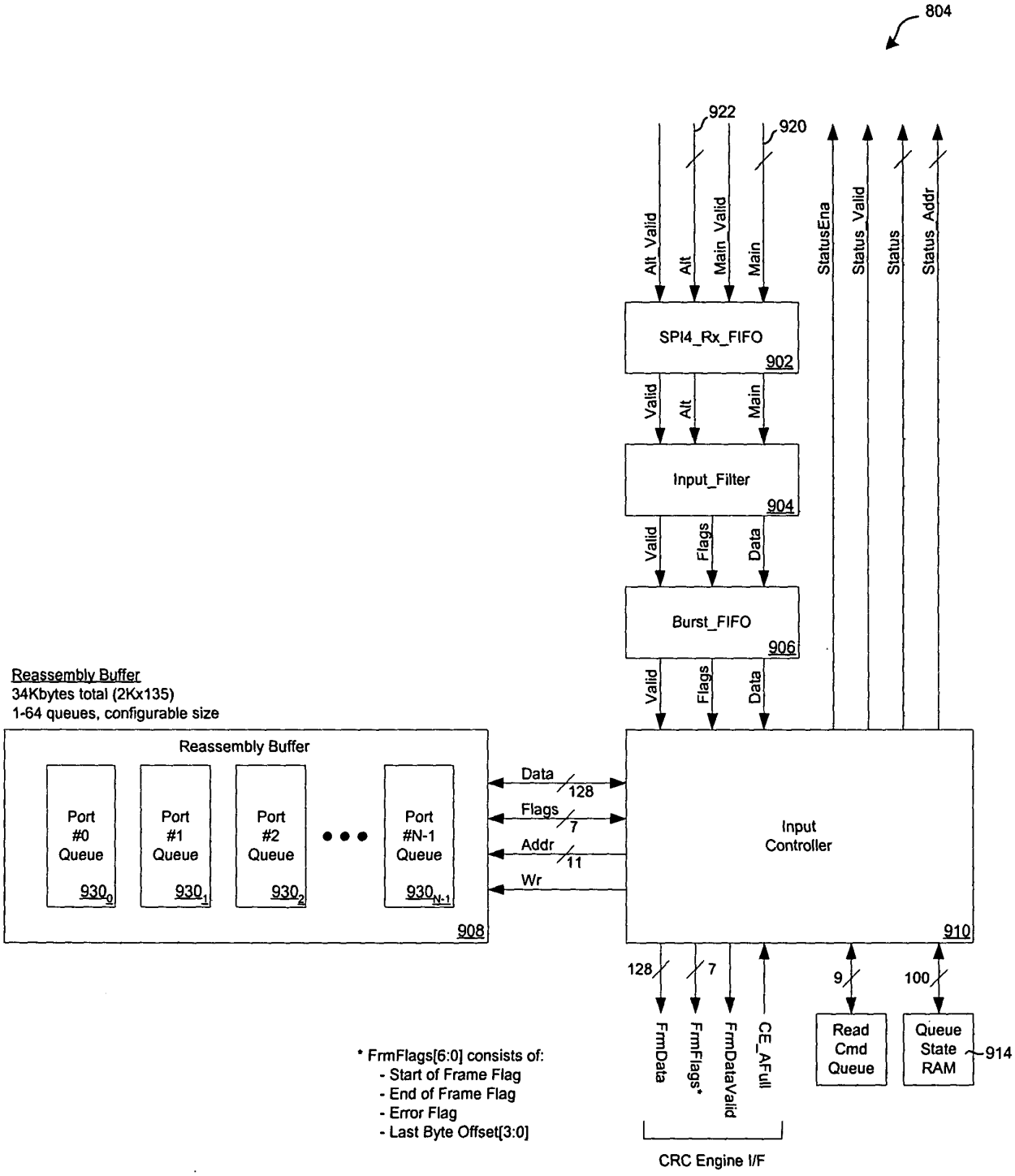

804
922
920
Alt_Valid
Alt
Main_Valid
Main
StatusEna
Status_Valid
Status
Status_Addr
SPI4_Rx_FIFO
902
Valid
Alt
Main
Input_Filter
904
Valid
Flags
Data
Burst_FIFO
906
Valid
Flags
Data
Reassembly Buffer
34Kbytes total (2Kx135)
1-64 queues, configurable size
Reassembly Buffer
Port #0 Queue
930_0
Port #1 Queue
930_1
Port #2 Queue
930_2
Port #N-1 Queue
930_N-1
908
Data
128
Flags
7
Addr
11
Wr
Input Controller
910
128
7
9
100
FrmData
FrmFlags*
FrmDataValid
CE_AFull
Read Cmd Queue
Queue State RAM
914
* FrmFlags[6:0] consists of:
- Start of Frame Flag
- End of Frame Flag
- Error Flag
- Last Byte Offset[3:0]
CRC Engine I/F

FIGURE 9

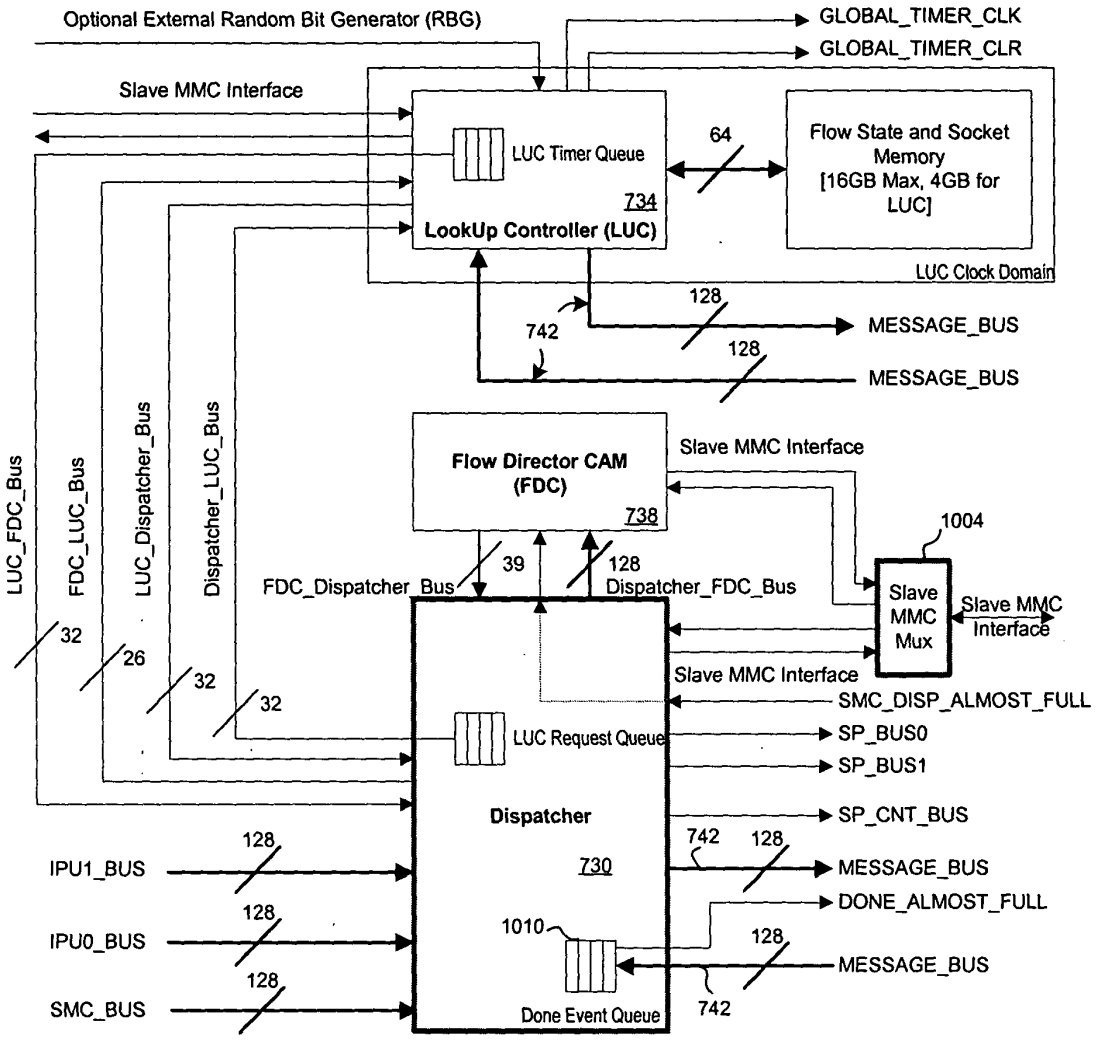

Optional External Random Bit Generator (RBG)
GLOBAL_TIMER_CLK
GLOBAL_TIMER_CLR
Slave MMC Interface
LUC Timer Queue
64
Flow State and Socket Memory [16GB Max, 4GB for LUC]
734
LookUp Controller (LUC)
LUC Clock Domain
128
742
MESSAGE_BUS
128
MESSAGE_BUS
Flow Director CAM (FDC)
738
Slave MMC Interface
1004
LUC_FDC_Bus
FDC_LUC_Bus
LUC_Dispatcher_Bus
Dispatcher_LUC_Bus
39
128
FDC_Dispatcher_Bus
Dispatcher_FDC_Bus
Slave MMC Mux
Slave MMC Interface
32
26
32
32
Slave MMC Interface
SMC_DISP_ALMOST_FULL
LUC Request Queue
SP_BUS0
SP_BUS1
Dispatcher
SP_CNT_BUS
730
742
128
MESSAGE_BUS
DONE_ALMOST_FULL
IPU1_BUS
128
IPU0_BUS
128
SMC_BUS
128
1010
128
MESSAGE_BUS
742
Done Event Queue

FIGURE 10

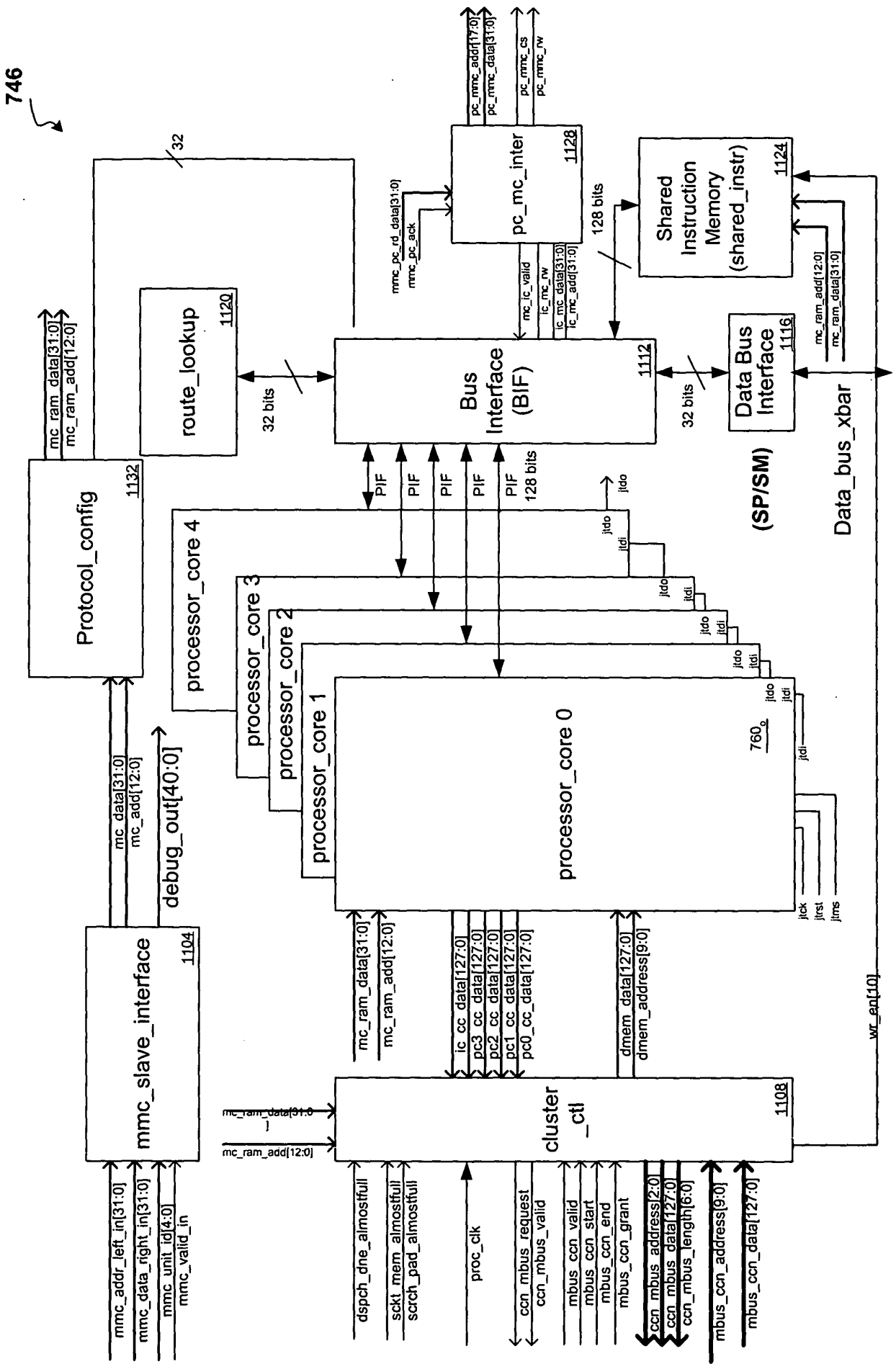

746
Protocol_config
1132
route_lookup
1120
Bus Interface (BIF)
1112
pc_mc_inter
1128
Shared Instruction Memory (shared_instr)
1124
Data Bus Interface
1116
Data_bus_xbar
(SP/SM)
processor_core 4
processor_core 3
processor_core 2
processor_core 1
processor_core 0
760_0
mmc_slave_interface
1104
cluster _ctl
1108
32
32 bits
128 bits
PIF
mc_ram_data[31:0]
mc_ram_add[12:0]
debug_out[40:0]
mc_data[31:0]
mc_add[12:0]
mmc_addr_left_in[31:0]
mmc_data_right_in[31:0]
mmc_unit_id[4:0]
mmc_valid_in
dspch_dne_almostfull
sckt_mem_almostfull
scrch_pad_almostfull
proc_clk
ccn_mbus_request
ccn_mbus_valid
mbus_ccn_valid
mbus_ccn_start
mbus_ccn_end
mbus_ccn_grant
mbus_ccn_address[9:0]
mbus_ccn_data[127:0]
ccn_mbus_address[2:0]
ccn_mbus_data[127:0]
ccn_mbus_length[6:0]
ic_cc_data[127:0]
pc3_cc_data[127:0]
pc2_cc_data[127:0]
pc1_cc_data[127:0]
pc0_cc_data[127:0]
dmem_data[127:0]
dmem_address[9:0]
wr_en[10]
jtdo
jtdi
jtck
jtrst
jtms
mmc_pc_rd_data[31:0]
mmc_pc_ack
mc_ic_valid
ic_mc_rw
ic_mc_data[31:0]
ic_mc_add[31:0]
pc_mmc_addr[17:0]
pc_mmc_data[31:0]
pc_mmc_cs
pc_mmc_rw
mc_ram_add[12:0]
mc_ram_data[31:0]

FIGURE 11

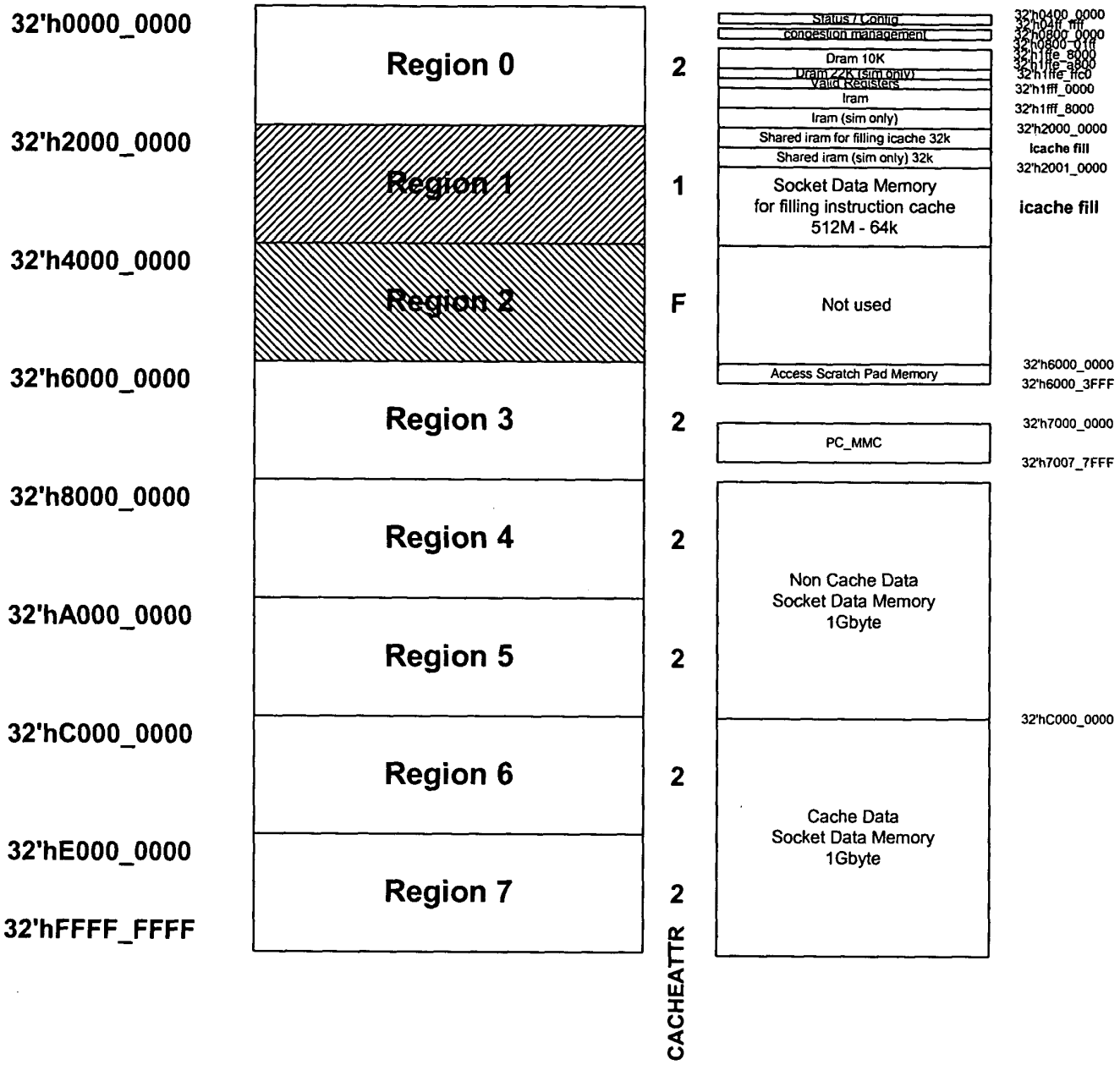

32'h0000_0000
Region 0
32'h2000_0000
Region 1
32'h4000_0000
Region 2
32'h6000_0000
Region 3
32'h8000_0000
Region 4
32'hA000_0000
Region 5
32'hC000_0000
Region 6
32'hE000_0000
Region 7
32'hFFFF_FFFF
2
1
F
2
2
2
2
2
CACHEATTR
Status / Config
congestion management
Dram 10K
Dram 22K (sim only)
Valid Registers
Iram
Iram (sim only)
Shared iram for filling icache 32k
Shared iram (sim only) 32k
Socket Data Memory
for filling instruction cache
512M - 64k
Not used
Access Scratch Pad Memory
PC_MMC
Non Cache Data
Socket Data Memory
1Gbyte
Cache Data
Socket Data Memory
1Gbyte
32'h0400_0000
32'h04ff_ffff
32'h0800_0000
32'h0800_01ff
32'h1ffe_8000
32'h1ffe_a800
32'h1ffe_ffc0
32'h1fff_0000
32'h1fff_8000
32'h2000_0000
icache fill
32'h2001_0000
icache fill
32'h6000_0000
32'h6000_3FFF
32'h7000_0000
32'h7007_7FFF
32'hC000_0000

FIGURE 12

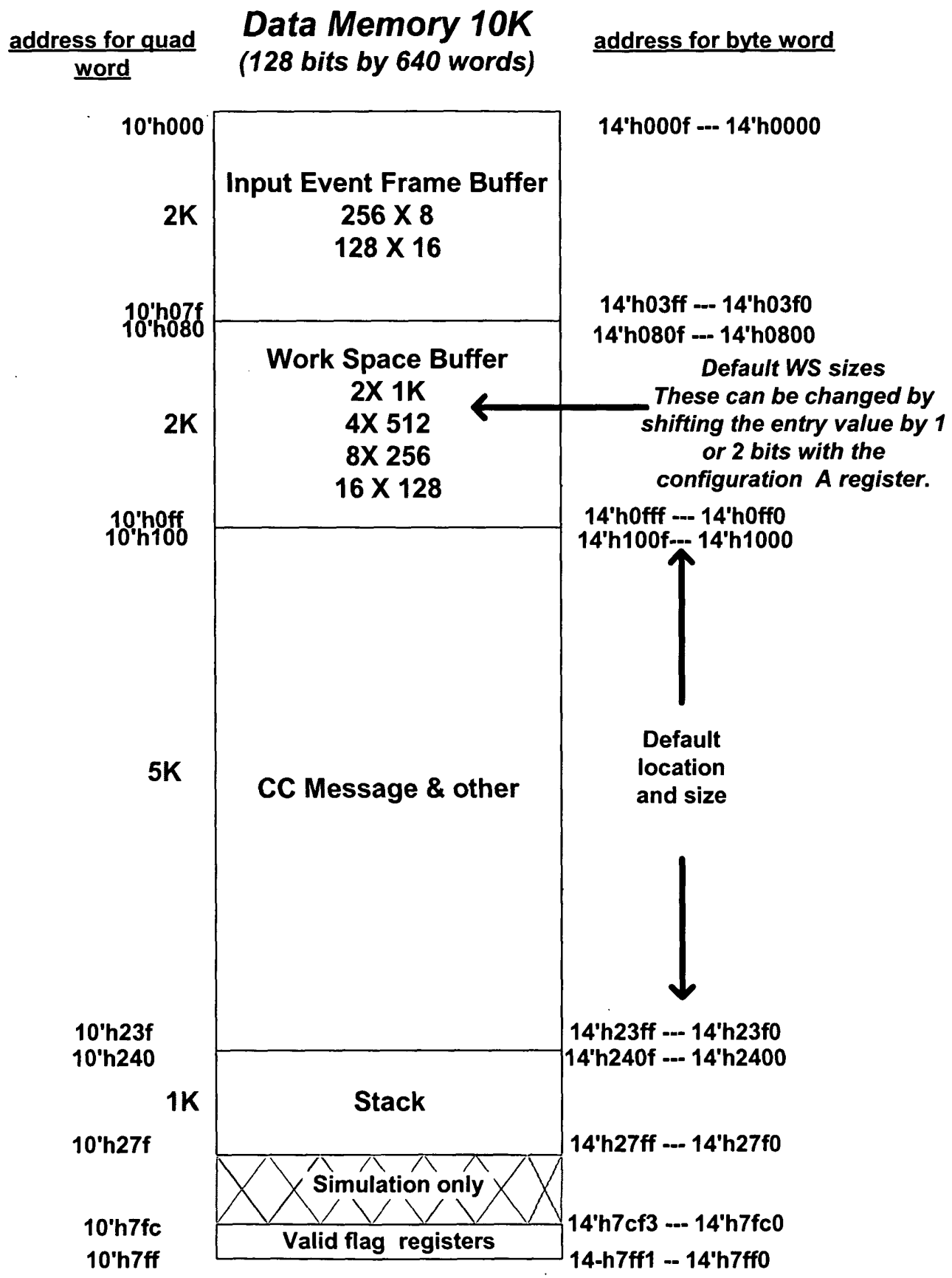

address for quad word
Data Memory 10K
(128 bits by 640 words)
address for byte word
10'h000
14'h000f --- 14'h0000
Input Event Frame Buffer
256 X 8
128 X 16
2K
10'h07f
14'h03ff --- 14'h03f0
10'h080
14'h080f --- 14'h0800
Work Space Buffer
2X 1K
4X 512
8X 256
16 X 128
2K
Default WS sizes
These can be changed by shifting the entry value by 1 or 2 bits with the configuration A register.
10'h0ff
14'h0fff --- 14'h0ff0
10'h100
14'h100f--- 14'h1000
Default location and size
5K
CC Message & other
10'h23f
14'h23ff --- 14'h23f0
10'h240
14'h240f --- 14'h2400
1K
Stack
10'h27f
14'h27ff --- 14'h27f0
Simulation only
10'h7fc
14'h7cf3 --- 14'h7fc0
Valid flag registers
10'h7ff
14-h7ff1 -- 14'h7ff0

FIGURE 13

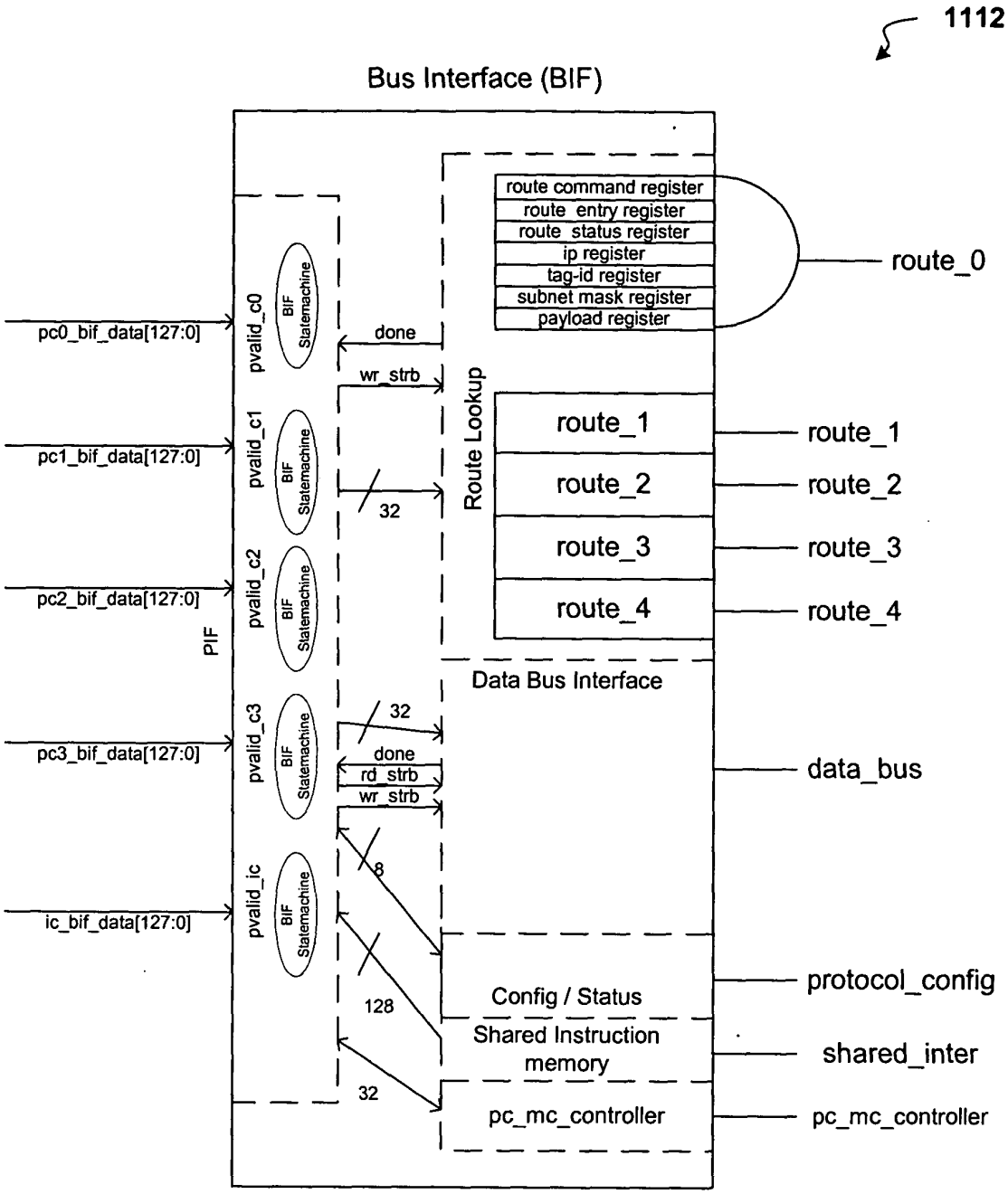

1112
Bus Interface (BIF)
route command register
route_entry register
route_status register
ip register
tag-id register
subnet mask register
payload register
route_0
pc0_bif_data[127:0]
pvalid_c0
BIF Statemachine
done
wr_strb
Route Lookup
route_1
route_1
pc1_bif_data[127:0]
pvalid_c1
BIF Statemachine
32
route_2
route_2
pc2_bif_data[127:0]
pvalid_c2
BIF Statemachine
route_3
route_3
PIF
route_4
route_4
Data Bus Interface
32
pc3_bif_data[127:0]
pvalid_c3
BIF Statemachine
done
rd_strb
wr_strb
data_bus
8
ic_bif_data[127:0]
pvalid_ic
BIF Statemachine
protocol_config
Config / Status
128
Shared Instruction memory
shared_inter
32
pc_mc_controller
pc_mc_controller

FIGURE 14

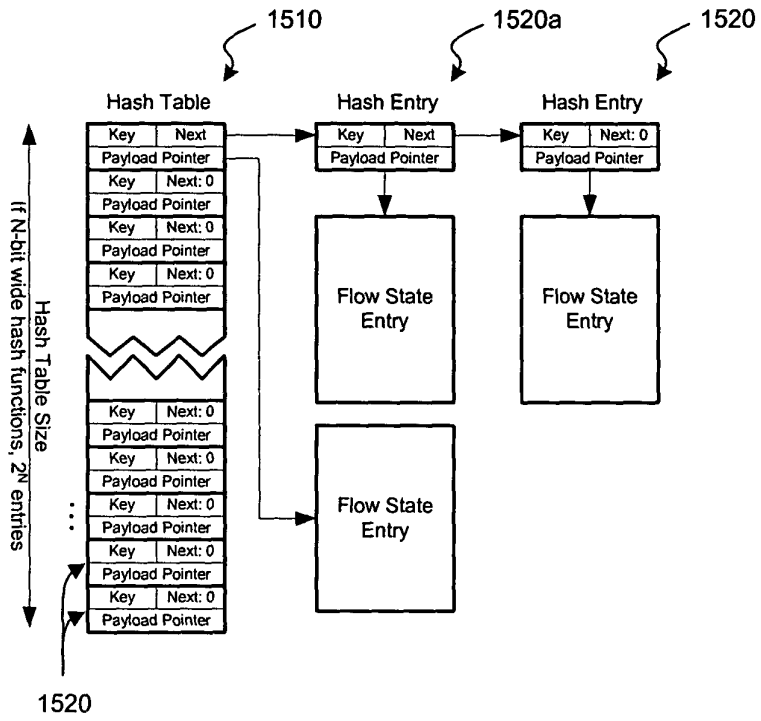

1510
1520a
1520
Hash Table
Hash Entry
Hash Entry
Key
Next
Payload Pointer
Next: 0
Flow State Entry
Hash Table Size
(If N-bit wide hash functions, 2^N entries)
1520

FIGURE 15

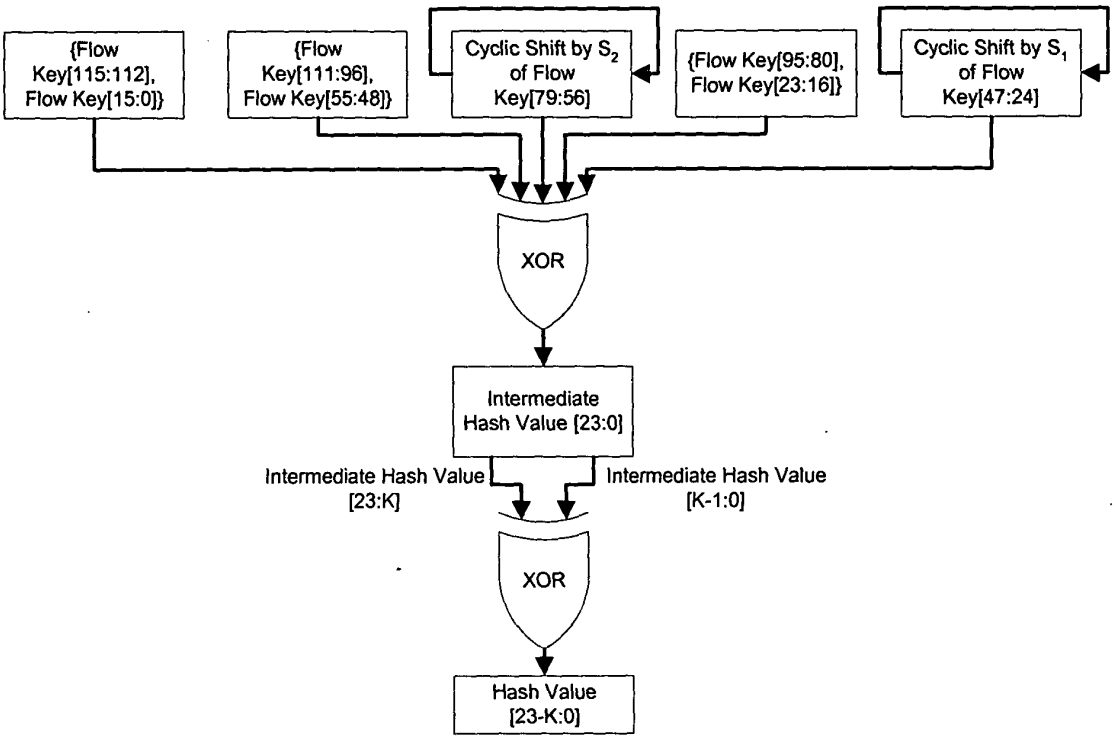

{Flow Key[115:112], Flow Key[15:0]}
{Flow Key[111:96], Flow Key[55:48]}
Cyclic Shift by S_2 of Flow Key[79:56]
{Flow Key[95:80], Flow Key[23:16]}
Cyclic Shift by S_1 of Flow Key[47:24]
XOR
Intermediate Hash Value [23:0]
Intermediate Hash Value [23:K]
Intermediate Hash Value [K-1:0]
XOR
Hash Value [23-K:0]

FIGURE 16

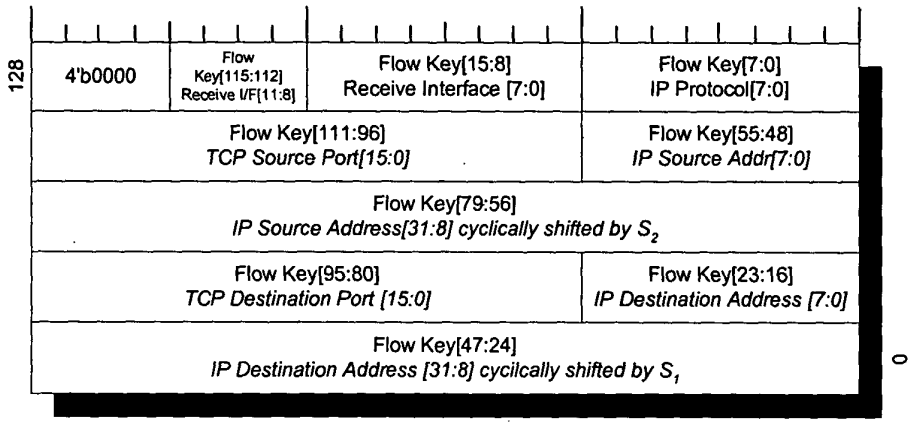

128
4'b0000
Flow Key[115:112] Receive I/F[11:8]
Flow Key[15:8] Receive Interface [7:0]
Flow Key[7:0] IP Protocol[7:0]
Flow Key[111:96] TCP Source Port[15:0]
Flow Key[55:48] IP Source Addr[7:0]
Flow Key[79:56] IP Source Address[31:8] cyclically shifted by S_2
Flow Key[95:80] TCP Destination Port [15:0]
Flow Key[23:16] IP Destination Address [7:0]
Flow Key[47:24] IP Destination Address [31:8] cycilcally shifted by S_1
0

FIGURE 17

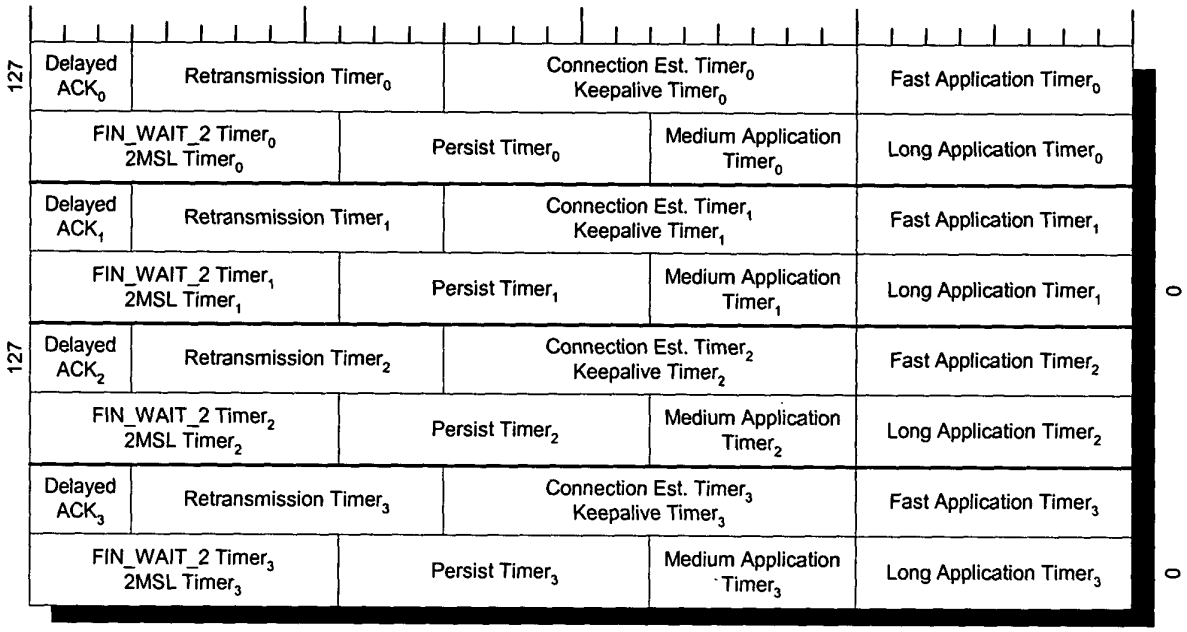

127
Delayed ACK0
Retransmission Timer0
Connection Est. Timer0
Keepalive Timer0
Fast Application Timer0
FIN_WAIT_2 Timer0
2MSL Timer0
Persist Timer0
Medium Application Timer0
Long Application Timer0
Delayed ACK1
Retransmission Timer1
Connection Est. Timer1
Keepalive Timer1
Fast Application Timer1
FIN_WAIT_2 Timer1
2MSL Timer1
Persist Timer1
Medium Application Timer1
Long Application Timer1
0
127
Delayed ACK2
Retransmission Timer2
Connection Est. Timer2
Keepalive Timer2
Fast Application Timer2
FIN_WAIT_2 Timer2
2MSL Timer2
Persist Timer2
Medium Application Timer2
Long Application Timer2
Delayed ACK3
Retransmission Timer3
Connection Est. Timer3
Keepalive Timer3
Fast Application Timer3
FIN_WAIT_2 Timer3
2MSL Timer3
Persist Timer3
Medium Application Timer3
Long Application Timer3
0

FIGURE 18

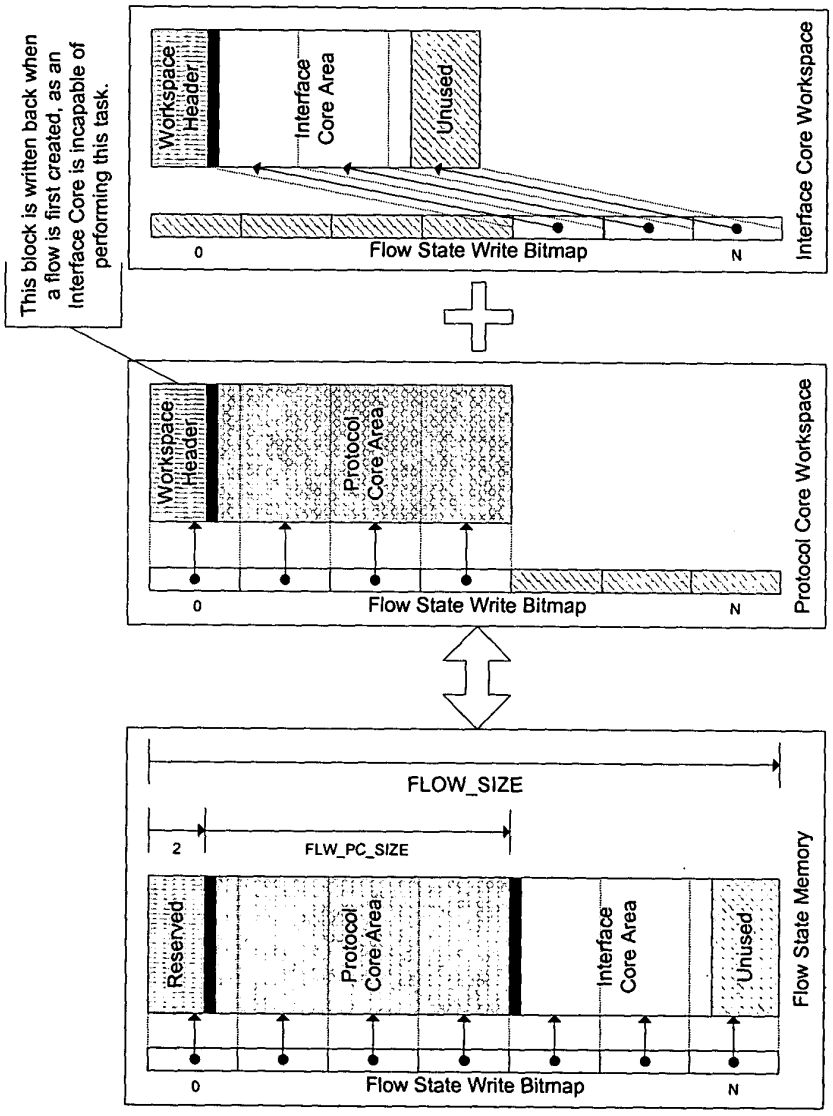

This block is written back when a flow is first created, as an Interface Core is incapable of performing this task.
Workspace Header
Interface Core Area
Unused
Interface Core Workspace
0
Flow State Write Bitmap
N
Workspace Header
Protocol Core Area
Protocol Core Workspace
0
Flow State Write Bitmap
N
FLOW_SIZE
2
FLW_PC_SIZE
Reserved
Protocol Core Area
Interface Core Area
Unused
Flow State Memory
0
Flow State Write Bitmap
N

FIGURE 19

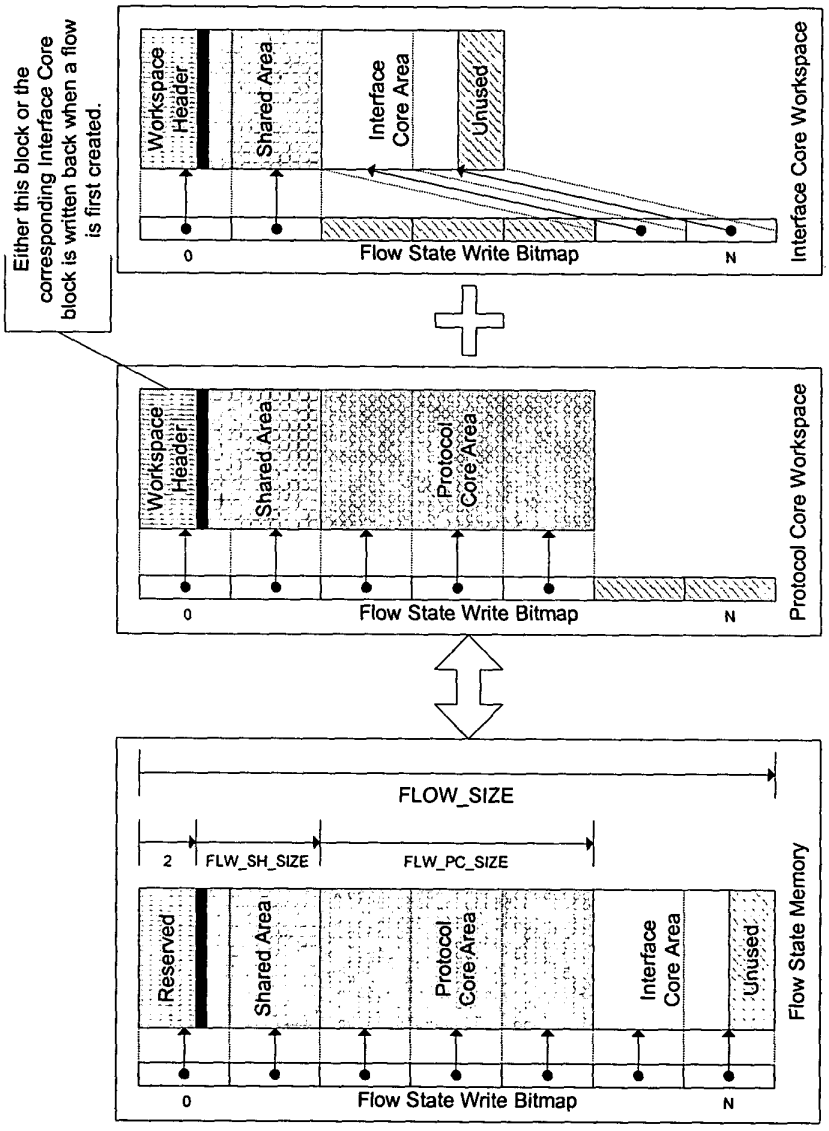

Either this block or the corresponding Interface Core block is written back when a flow is first created.
Workspace Header
Shared Area
Interface Core Area
Unused
Flow State Write Bitmap
0
N
Interface Core Workspace
Workspace Header
Shared Area
Protocol Core Area
Flow State Write Bitmap
0
N
Protocol Core Workspace
FLOW_SIZE
2
FLW_SH_SIZE
FLW_PC_SIZE
Reserved
Shared Area
Protocol Core Area
Interface Core Area
Unused
Flow State Write Bitmap
0
N
Flow State Memory

FIGURE 20

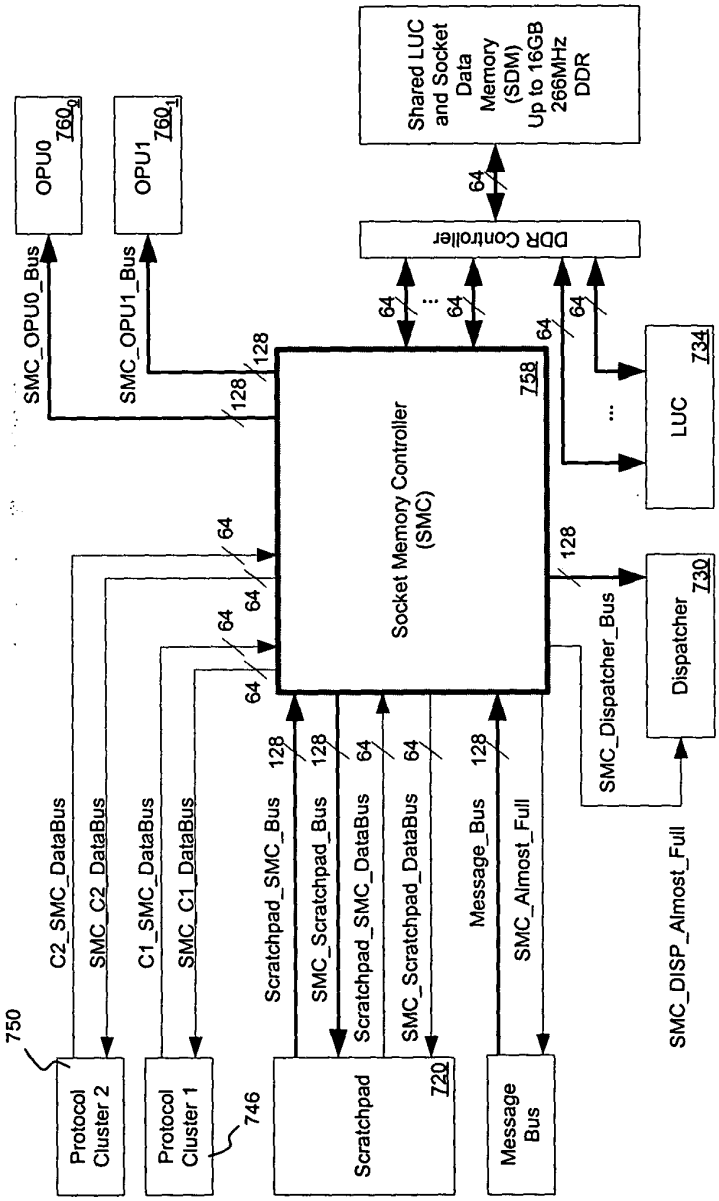

Shared LUC and Socket Data Memory (SDM) Up to 16GB 266MHz DDR
DDR Controller
OPU0
760₀
OPU1
760₁
SMC_OPU0_Bus
SMC_OPU1_Bus
Socket Memory Controller (SMC)
758
LUC
734
Dispatcher
730
SMC_Dispatcher_Bus
SMC_DISP_Almost_Full
C2_SMC_DataBus
SMC_C2_DataBus
C1_SMC_DataBus
SMC_C1_DataBus
Scratchpad_SMC_Bus
SMC_Scratchpad_Bus
Scratchpad_SMC_DataBus
SMC_Scratchpad_DataBus
Message_Bus
SMC_Almost_Full
Protocol Cluster 2
Protocol Cluster 1
Scratchpad
720
Message Bus
750
746
64
128

FIGURE 21

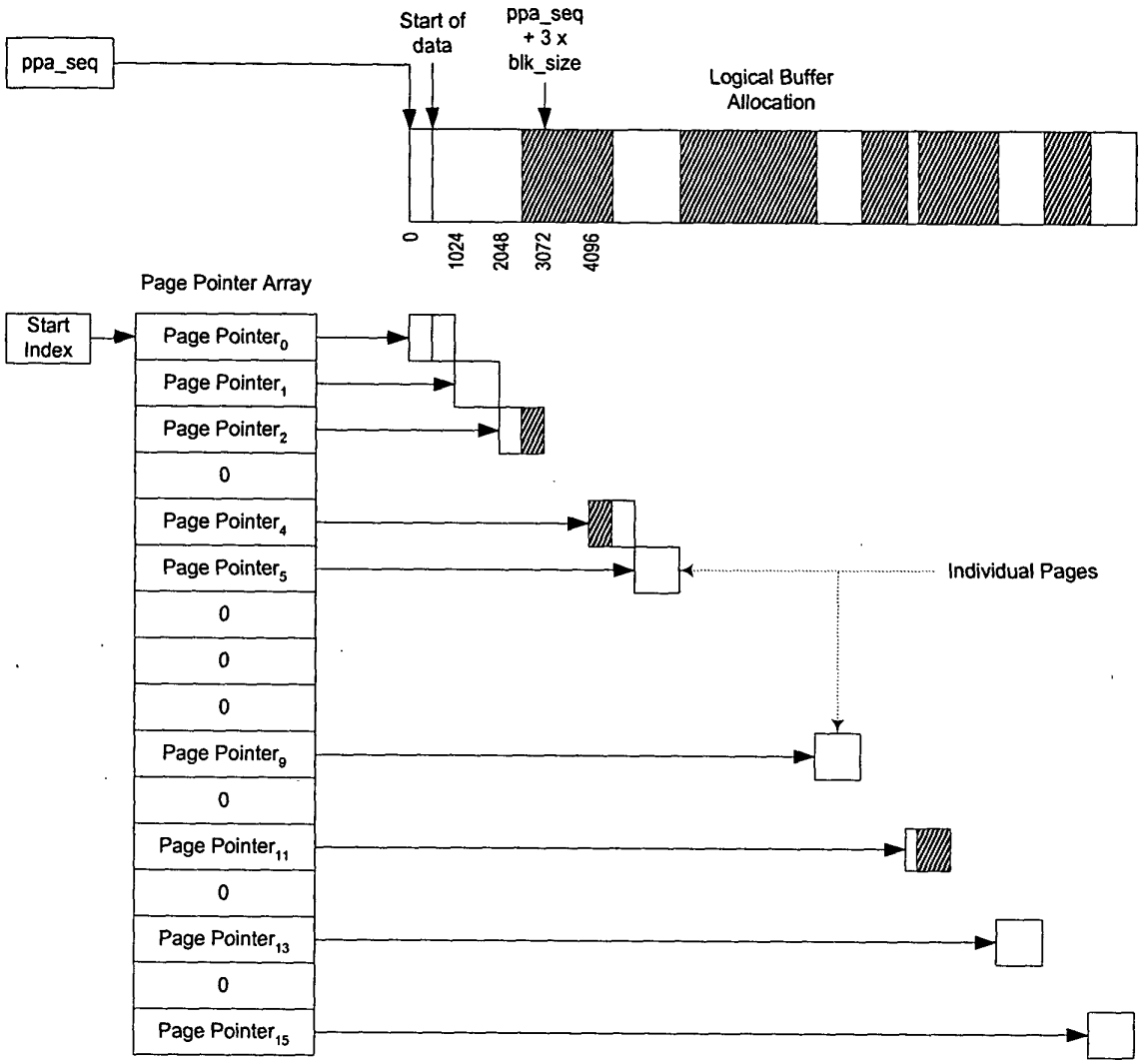

ppa_seq
Start of data
ppa_seq + 3 x blk_size
Logical Buffer Allocation
0
1024
2048
3072
4096
Page Pointer Array
Start Index
Page Pointer0
Page Pointer1
Page Pointer2
0
Page Pointer4
Page Pointer5
Individual Pages
0
0
0
Page Pointer9
0
Page Pointer11
0
Page Pointer13
0
Page Pointer15

FIGURE 22

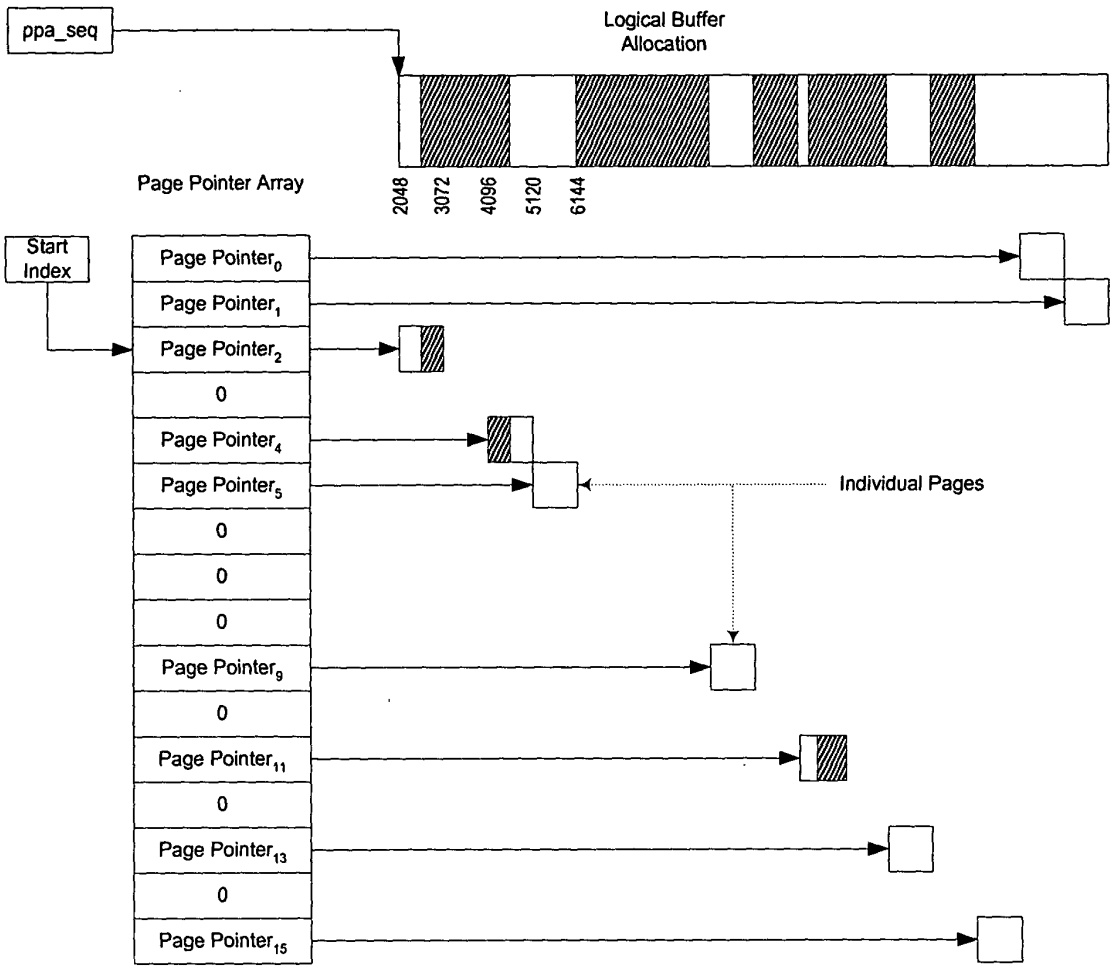

ppa_seq
Logical Buffer Allocation
Page Pointer Array
2048
3072
4096
5120
6144
Start Index
Page Pointer0
Page Pointer1
Page Pointer2
0
Page Pointer4
Page Pointer5
Individual Pages
0
0
0
Page Pointer9
0
Page Pointer11
0
Page Pointer13
0
Page Pointer15

FIGURE 23

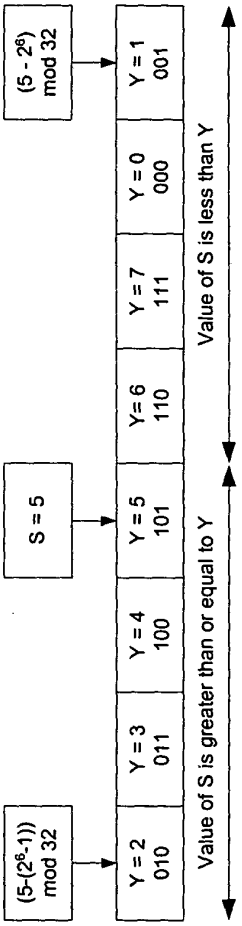


FIGURE 24

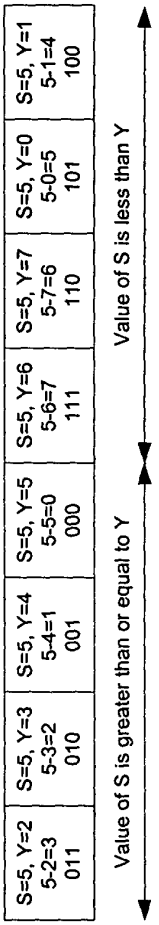


FIGURE 25

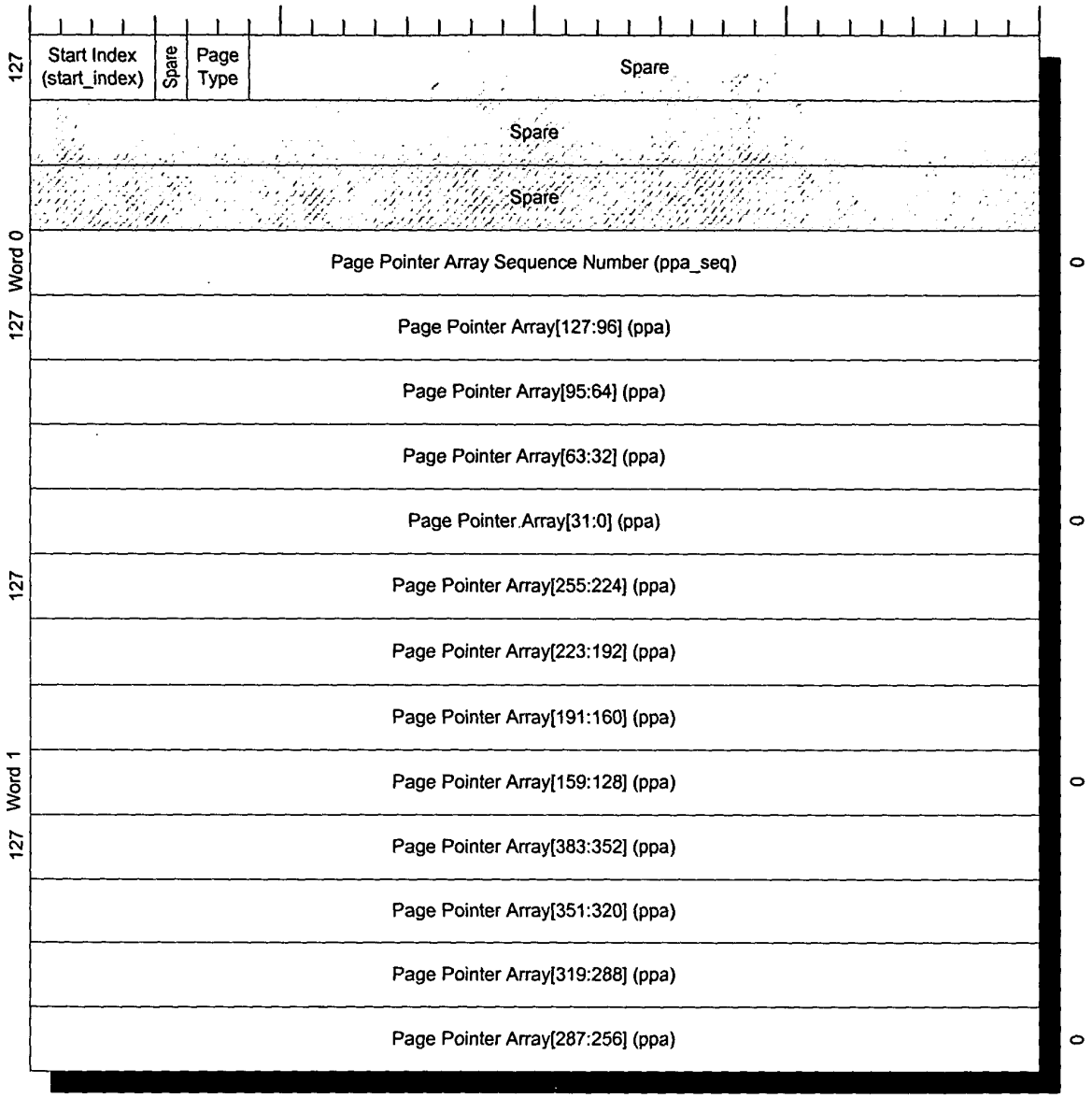

127
Start Index (start_index)
Spare
Page Type
Spare
Spare
Spare
Word 0
127
Page Pointer Array Sequence Number (ppa_seq)
0
Page Pointer Array[127:96] (ppa)
Page Pointer Array[95:64] (ppa)
Page Pointer Array[63:32] (ppa)
Page Pointer Array[31:0] (ppa)
0
127
Page Pointer Array[255:224] (ppa)
Page Pointer Array[223:192] (ppa)
Page Pointer Array[191:160] (ppa)
Word 1
Page Pointer Array[159:128] (ppa)
0
127
Page Pointer Array[383:352] (ppa)
Page Pointer Array[351:320] (ppa)
Page Pointer Array[319:288] (ppa)
Page Pointer Array[287:256] (ppa)
0

FIGURE 26

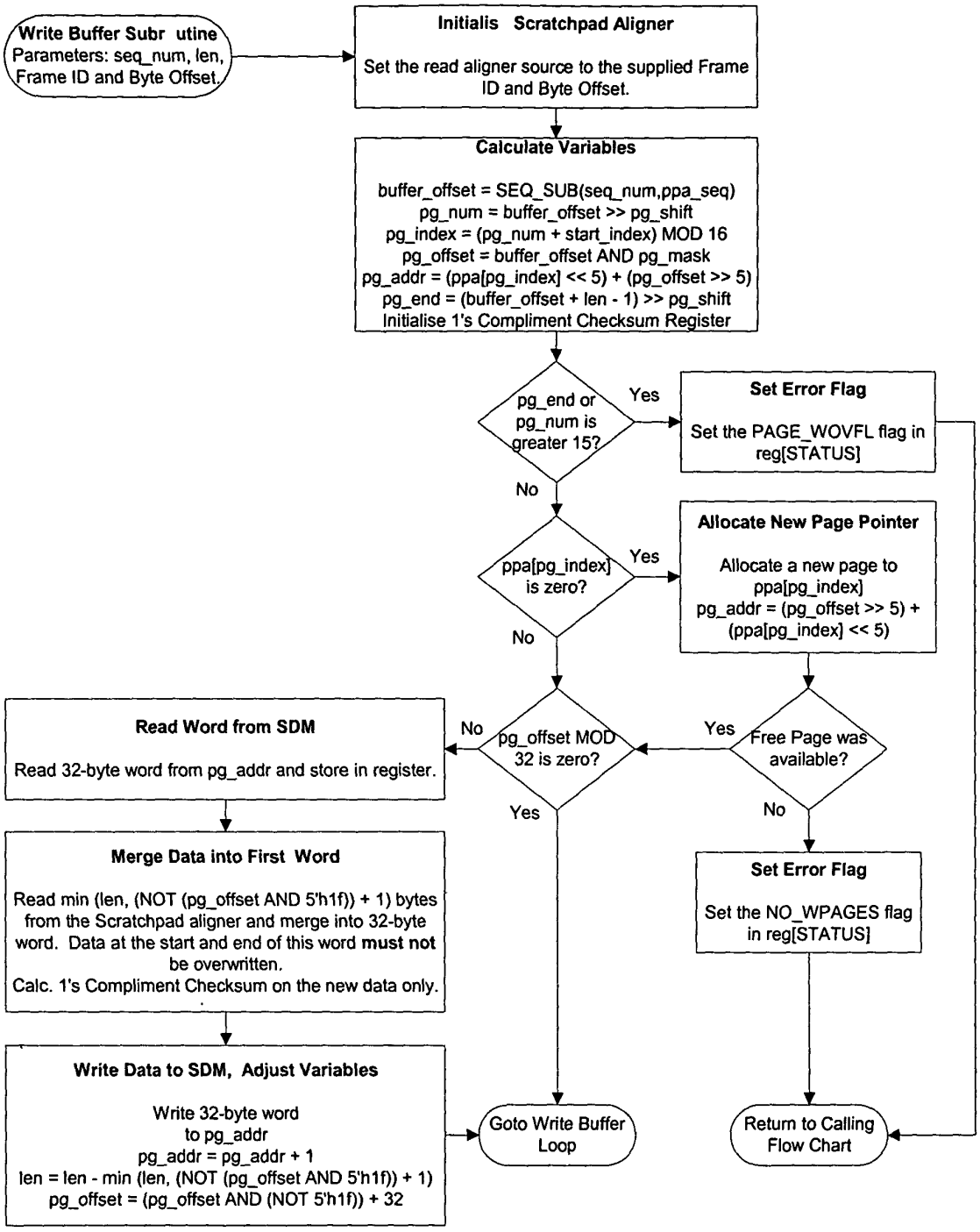

Write Buffer Subr utine
Parameters: seq_num, len, Frame ID and Byte Offset.
Initialis Scratchpad Aligner
Set the read aligner source to the supplied Frame ID and Byte Offset.
Calculate Variables
buffer_offset = SEQ_SUB(seq_num,ppa_seq)
pg_num = buffer_offset >> pg_shift
pg_index = (pg_num + start_index) MOD 16
pg_offset = buffer_offset AND pg_mask
pg_addr = (ppa[pg_index] << 5) + (pg_offset >> 5)
pg_end = (buffer_offset + len - 1) >> pg_shift
Initialise 1's Compliment Checksum Register
pg_end or pg_num is greater 15?
Yes
Set Error Flag
Set the PAGE_WOVFL flag in reg[STATUS]
No
ppa[pg_index] is zero?
Yes
Allocate New Page Pointer
Allocate a new page to ppa[pg_index]
pg_addr = (pg_offset >> 5) + (ppa[pg_index] << 5)
No
Read Word from SDM
Read 32-byte word from pg_addr and store in register.
No
pg_offset MOD 32 is zero?
Yes
Free Page was available?
Yes
No
Merge Data into First Word
Read min (len, (NOT (pg_offset AND 5'h1f)) + 1) bytes from the Scratchpad aligner and merge into 32-byte word. Data at the start and end of this word must not be overwritten.
Calc. 1's Compliment Checksum on the new data only.
Set Error Flag
Set the NO_WPAGES flag in reg[STATUS]
Write Data to SDM, Adjust Variables
Write 32-byte word to pg_addr
pg_addr = pg_addr + 1
len = len - min (len, (NOT (pg_offset AND 5'h1f)) + 1)
pg_offset = (pg_offset AND (NOT 5'h1f)) + 32
Goto Write Buffer Loop
Return to Calling Flow Chart

FIGURE 27

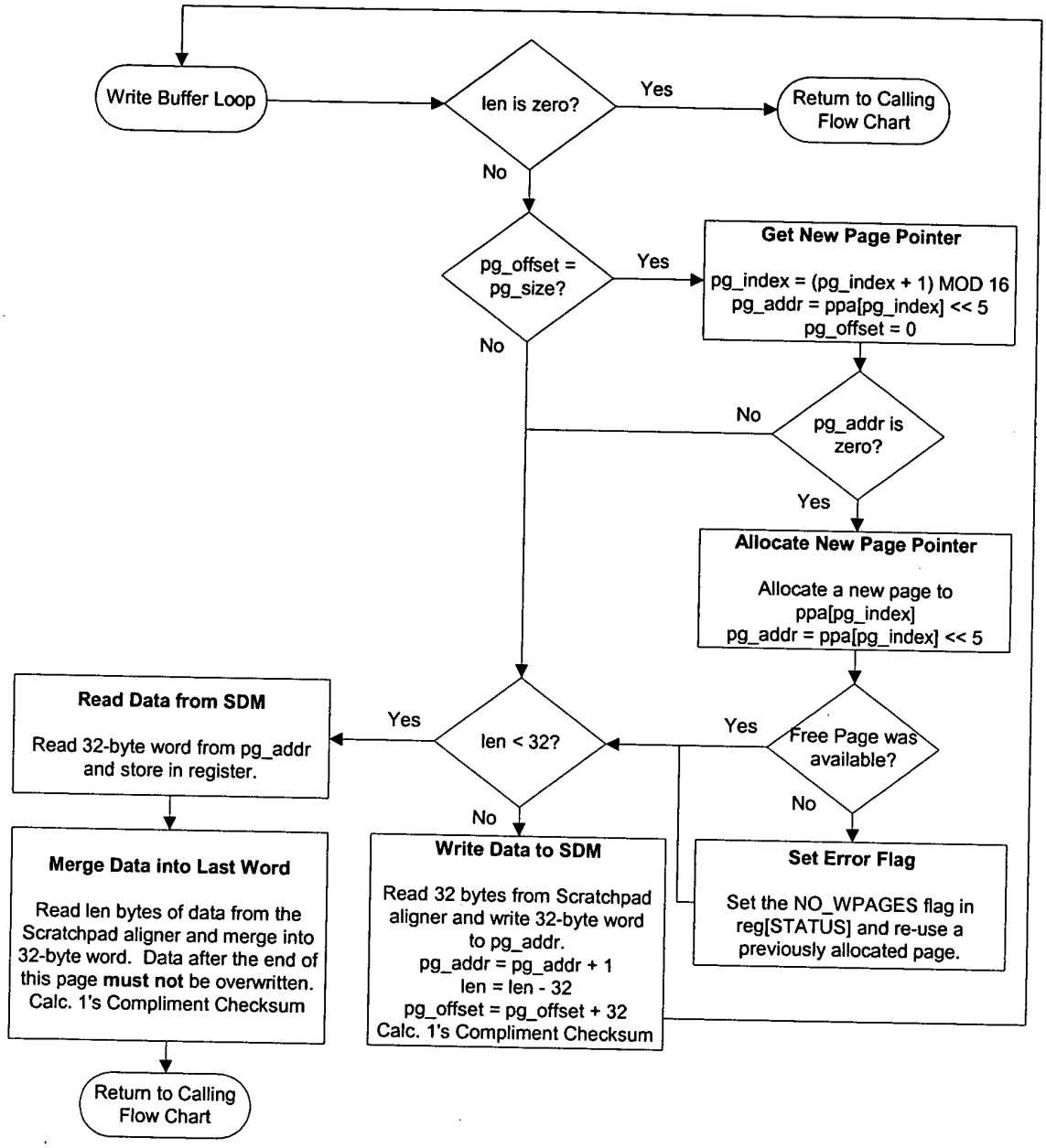

Write Buffer Loop
len is zero?
Yes
Return to Calling Flow Chart
No
pg_offset = pg_size?
Yes
Get New Page Pointer
pg_index = (pg_index + 1) MOD 16
pg_addr = ppa[pg_index] << 5
pg_offset = 0
No
No
pg_addr is zero?
Yes
Allocate New Page Pointer
Allocate a new page to ppa[pg_index]
pg_addr = ppa[pg_index] << 5
Read Data from SDM
Read 32-byte word from pg_addr and store in register.
Yes
len < 32?
Yes
Free Page was available?
No
No
Write Data to SDM
Read 32 bytes from Scratchpad aligner and write 32-byte word to pg_addr.
pg_addr = pg_addr + 1
len = len - 32
pg_offset = pg_offset + 32
Calc. 1's Compliment Checksum
Set Error Flag
Set the NO_WPAGES flag in reg[STATUS] and re-use a previously allocated page.
Merge Data into Last Word
Read len bytes of data from the Scratchpad aligner and merge into 32-byte word. Data after the end of this page must not be overwritten.
Calc. 1's Compliment Checksum
Return to Calling Flow Chart

FIGURE 28

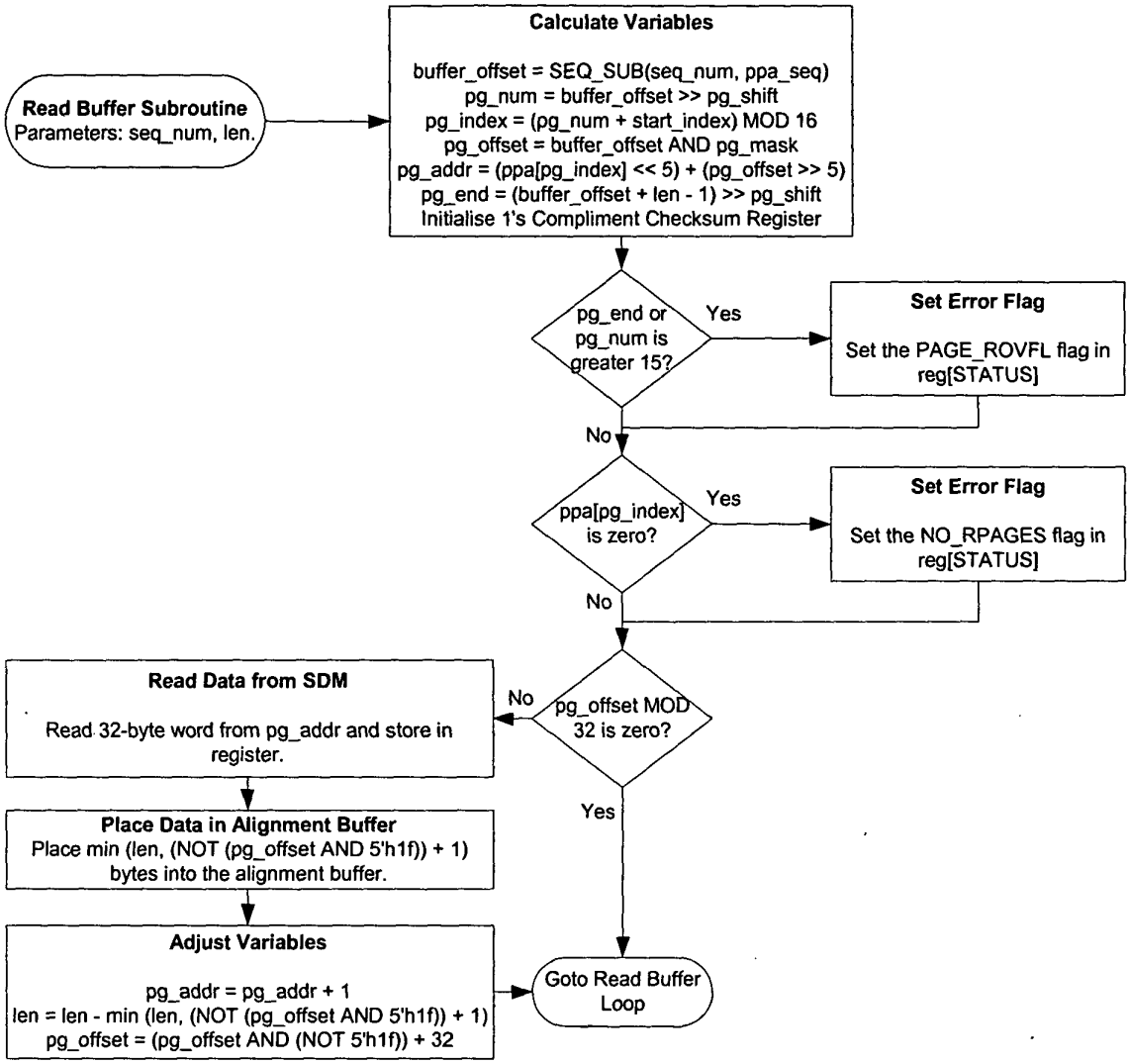

Read Buffer Subroutine
Parameters: seq_num, len.
Calculate Variables
buffer_offset = SEQ_SUB(seq_num, ppa_seq)
pg_num = buffer_offset >> pg_shift
pg_index = (pg_num + start_index) MOD 16
pg_offset = buffer_offset AND pg_mask
pg_addr = (ppa[pg_index] << 5) + (pg_offset >> 5)
pg_end = (buffer_offset + len - 1) >> pg_shift
Initialise 1's Compliment Checksum Register
pg_end or pg_num is greater 15?
Yes
Set Error Flag
Set the PAGE_ROVFL flag in reg[STATUS]
No
ppa[pg_index] is zero?
Yes
Set Error Flag
Set the NO_RPAGES flag in reg[STATUS]
No
pg_offset MOD 32 is zero?
No
Read Data from SDM
Read 32-byte word from pg_addr and store in register.
Yes
Place Data in Alignment Buffer
Place min (len, (NOT (pg_offset AND 5'h1f)) + 1) bytes into the alignment buffer.
Adjust Variables
pg_addr = pg_addr + 1
len = len - min (len, (NOT (pg_offset AND 5'h1f)) + 1)
pg_offset = (pg_offset AND (NOT 5'h1f)) + 32
Goto Read Buffer Loop

FIGURE 29

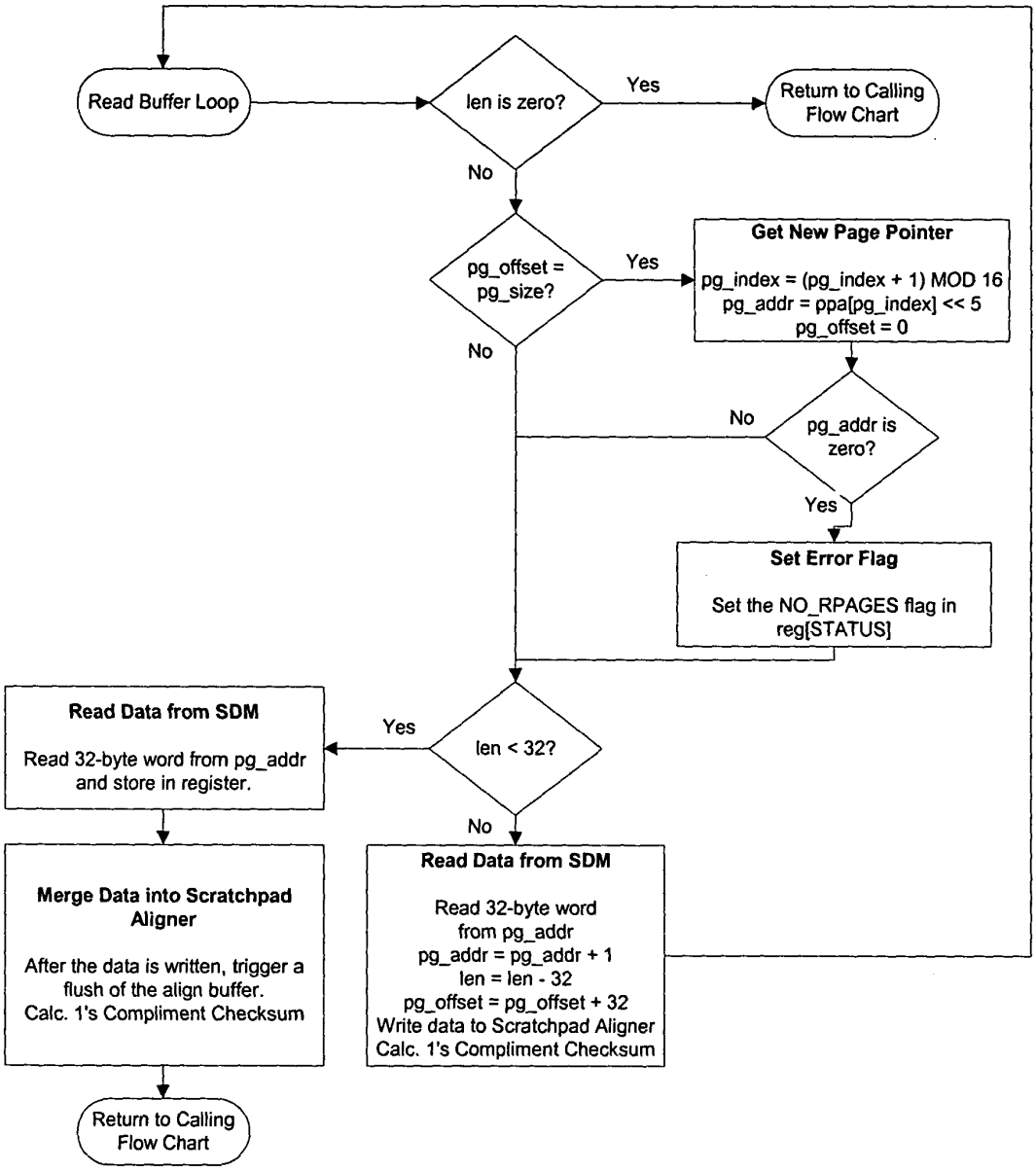

Read Buffer Loop
len is zero?
Yes
Return to Calling Flow Chart
No
pg_offset = pg_size?
Yes
Get New Page Pointer
pg_index = (pg_index + 1) MOD 16
pg_addr = ppa[pg_index] << 5
pg_offset = 0
No
No
pg_addr is zero?
Yes
Set Error Flag
Set the NO_RPAGES flag in reg[STATUS]
Read Data from SDM
Read 32-byte word from pg_addr and store in register.
Yes
len < 32?
No
Merge Data into Scratchpad Aligner
After the data is written, trigger a flush of the align buffer.
Calc. 1's Compliment Checksum
Read Data from SDM
Read 32-byte word from pg_addr
pg_addr = pg_addr + 1
len = len - 32
pg_offset = pg_offset + 32
Write data to Scratchpad Aligner
Calc. 1's Compliment Checksum
Return to Calling Flow Chart

FIGURE 30